Olimpiada Societății pentru Excelență și Performanță în Informatică

ETAPA NAȚIONALĂ GIMNAZIU
BARAJ JUNIORI 2021

SUBIECTE

Enunțuri și rezolvări

2021 - Editura L&S Soft | Infobits Academy
https://ebooks.infobits.ro

ISBN: 978-606-95233-1-5

M A T E R I A L U L E S T E D I S T R I B U I T G R A T U I T .
S P O R L A S T U D I U !

Editura L&S SOFT

Adresa: Aleea Aviației nr. 10, Voluntari, Ilfov
RO14864546, J23/1969/2019
Mobil: 0727.731.947
E-mail: hello@infobits.ro

www.manuale-de-informatica.ro

Infobits Academy
www.infobits.ro

Cărți în format electronic cu specific informatic
https://ebooks.infobits.ro

Autori

Clasa a V-a

- ☺ prof. Florentina Ungureanu - Colegiul Național de Informatică Piatra-Neamț (coordonator)
- ☺ prof. Georgeta Iulia Balacea - Colegiul Național "Vasile Alecsandri"
- ☺ prof. Dan Octavian Dumitrașcu - Colegiul Național Dinicu Golescu Câmpulung-Muscel
- ☺ prof. Violeta - Marilena Grecea - Colegiul Național de Informatică "Matei Basarab" Râmnicu-Vâlcea
- ☺ prof. Marius Nicoli - Colegiul National "Frații Buzești" Craiova
- ☺ prof. Adriana Simulescu - Liceul Teoretic "Grigore Moisil" Timișoara
- ☺ olimpic internațional (2003 - 2006) Dan-Constantin Spătărel

Clasa a VI-a

- ☺ prof. Daniela Lica, Centrul Județean de Excelență Prahova, Ploiești (coordonator)
- ☺ prof. Cristina Anton, Colegiul Național "Gheorghe Munteanu Murgoci", Brăila
- ☺ prof. Ana-Maria Arișanu, Colegiul Național "Mircea cel Bătrân", Rm. Vâlcea
- ☺ stud. Stelian Chichirim, Universitatea București
- ☺ prof. Cristina Iordaiche, Liceul Teoretic "Grigore Moisil", Timișoara
- ☺ stud. masterand Radu Muntean, Master CS - Zürich
- ☺ prof. Marinel Șerban, Colegiul Național "Emil Racoviță", Iași
- ☺ prof. Roxana Tîmplaru, Liceul Tehnologic "Ștefan Odobleja"/ ISJ Dolj

Clasa a VII-a

- ☺ prof. Emanuela Cerchez - Colegiul Naţional "Emil Racoviţă", Iaşi (coordonator)
- ☺ prof. Nistor Moţ - Şcoala "Dr. Luca", Brăila
- ☺ prof. Marius Nicoli - Colegiul Național "Frații Buzești", Craiova
- ☺ prof. Adrian Pintea - Colegiul Național "Andrei Mureşanu", Dej
- ☺ prof. Ionel-Vasile Piţ-Rada - Colegiul Naţional "Traian", Drobeta-Turnu Severin
- ☺ stud. Theodor-Gabriel Tulbă-Lecu - Universitatea Politehnica București

Clasa a VIII-a

- ☺ prof. Carmen Mincă - Colegiul Național de Informatică “Tudor Vianu”, București (coordonator)
- ☺ ș.l. Stelian Ciurea - Universitatea ”Lucian Blaga” din Sibiu
- ☺ prof. Flavius Dumitru Boian - Colegiul Național “Spiru Haret”, Târgu-Jiu
- ☺ prof. Veronica Raluca Costineanu - Colegiul Național “Ștefan cel Mare”, Suceava
- ☺ stud.drd. Diana Ghinea - Eidgenössische Technische Hochschule (ETH) Zürich
- ☺ stud.masterand Bogdan Ioan Iordache - Universitatea din București
- ☺ prof. Mirela Mlisan - Colegiul Național “Mircea cel Bătrân”, Râmnicu Vâlcea
- ☺ prof. Mircea Dorin Rotar - Colegiul Național “Samuil Vulcan”, Beiuș
- ☺ stud. Ioan Cristian Pop - Universitatea Politehnică București

Baraj Juniori

- ☺ prof. Marius Nicoli - Colegiul Național ”Frații Buzești” Craiova (coordonator)
- ☺ prof. Daniela Lica - Centrul Județean de Excelență Prahova, Ploiești
- ☺ prof. Flavius Dumitru Boian - Colegiul Național “Spiru Haret”, Târgu-Jiu
- ☺ prof. Mihai Bunget - Colegiul Național ”Tudor Vladimirescu”, Târgu Jiu
- ☺ prof. Mihai Bolohan - Liceul ”Regina Maria”, Dorohoi, Botoșani
- ☺ prof. Ciprian Chește - Liceul Tehnologic ”Grigore C. Moisil”, Buzău
- ☺ prof. Veronica Raluca Costineanu - Colegiul Național “Ștefan cel Mare”, Suceava
- ☺ prof. Dan Octavian Dumitrașcu - Colegiul Național, Dinicu Golescu Câmpulung-Muscel
- ☺ prof. Cristina Iordaiche - Liceul Teoretic “Grigore Moisil“, Timișoara
- ☺ prof. Eugen Nodea - Colegiul Național ”Tudor Vladimirescu”, Târgu Jiu
- ☺ prof. Ionel-Vasile Piţ-Rada - Colegiul Naţional ”Traian”, Drobeta-Turnu Severin
- ☺ prof. Petru Simion Opriță - Liceul ”Regina Maria”, Dorohoi, Botoșani
- ☺ prof. Dan Pracsiu - Liceul Teoretic ”Emil Racoviță”, Vaslui
- ☺ stud. Ioan Cristian Pop - Universitatea Politehnică București
- ☺ stud. Theodor-Gabriel Tulbă-Lecu, Universitatea Politehnică București

Cuprins

Capitolul 1

Clasa a V-a

1.1 Problema ktlon

Propunător: prof Florentina Ungureanu
Colegiul Național de Informatică Piatra-Neamț

Enunț

Două echipe, F și R, formate din n jucători fiecare, au participat în cadrul noii ediții ktlon la k probe. După fiecare probă s-au înregistrat în registrul ktlon 2*n valori: primele n reprezintă numărul de puncte câștigate în cadrul probei de jucătorii echipei F și următoarele n reprezintă numărul de puncte câștigate în cadrul probei de jucătorii echipei R.

Pentru ca o echipă să câștige o probă este necesar ca cel puțin unul din jucătorii săi să obțină un număr de puncte strict mai mare decât fiecare din punctajele obținute de către jucătorii celeilalte echipe.

Echipa câștigătoare a probei primește un număr de stele. Pentru a stabili numărul de stele primite, mai întâi se determină numărul M de jucători care au obținut un număr de puncte strict mai mare decât fiecare din punctajele obținute de jucătorii celeilalte echipe. Apoi echipa câștigătoare primește un număr de stele egal cu diferența dintre suma celor mai mari M punctaje obținute de jucătorii echipei câștigătoare și suma celor mai mari M punctaje obținute de jucătorii celeilalte echipe.

De exemplu, dacă jucătorii celor două echipe au obținut punctajele (8, 5, 8, 3, 9, 7) și $(5, 7, 5, 4, 5, 1)$, atunci $M = 3$ deoarece trei punctaje ale jucătorilor echipei F (8, 8, 9) sunt mai mari decât toate punctajele obținute de jucătorii echipei R. Echipa F câștigă proba și primește 8 stele $= (9 + 8 + 8) - (7 + 5 + 5)$.

Dacă niciun jucător al niciunei echipe nu obține un număr de puncte strict mai mare decât toate punctajele obținute de jucătorii celeilalte echipe, proba se încheie cu remiză și nicio echipă nu primește nicio stea ($M = 0$).

Competiția este câștigată de echipa care acumulează un număr maxim de stele la finalul tuturor probelor.

Cerință

Cunoscând n - numărul de jucători din fiecare echipă, k - numărul de probe și pentru fiecare probă punctajele obținute de cei 2*n jucători ai celor două echipe, determinați:

1. numărul de probe câștigate de echipa R;
2. numărul de stele obținut de echipa câștigătoare.

Date de intrare

Fișierul de intrare *ktlon.in* conține pe prima linie un număr C reprezentând cerința care trebuie să fie rezolvată (1 sau 2). Pe a doua linie se află două numere naturale n și k, care reprezintă numărul de jucători ai fiecărei echipe, respectiv numărul de probe, iar pe fiecare din următoarele k linii, câte 2*n numere naturale: primele n reprezintă numărul de puncte câștigate în cadrul probei curente de jucătorii echipei F și următoarele n reprezintă numărul de puncte câștigate în cadrul probei curente de jucătorii echipei R.

Numerele de pe aceeași linie sunt separate prin câte un spațiu.

Date de ieșire

Dacă $C = 1$, fișierul de ieșire *ktlon.out* va conține numărul de probe câștigate de echipa R.

Dacă $C = 2$, fișierul de ieșire va conține numărul de stele obținute de echipa câștigătoare.

Restricții și precizări

- $1 \leq C \leq 2$.
- $1 \leq n \leq 10\ 000$.
- $1 \leq k \leq 50$.
- Punctajele obținute de concurenți sunt numere naturale cuprinse între 0 și 200 000 inclusiv.
- Pentru teste valorând 35 de puncte cerința este 1.
- Pentru teste valorând 30 de puncte cerința este 2 și se garantează că $0 \leq M \leq 1$ pentru toate probele aceluiași test.
- Pentru teste valorând 35 de puncte cerința este 2 și se garantează că $0 \leq M \leq 5$ pentru toate probele aceluiași test.

Exemple

	ktlon.in	ktlon.out	**Explicații**
1)	1 3 4 6 8 3 7 7 6 1 2 3 4 5 3 1 5 3 4 5 2 1 5 3 4 5 2	1	Se rezolvă cerința **1**. Prima probă este câștigată de echipa F deoarece există un jucător care a obținut mai multe puncte (8) decât numărul de puncte câștigat de fiecare din jucătorii echipei R (7, 7, 6). A doua probă este câștigată de echipa R deoarece există doi jucători care au obținut mai multe puncte (4, 5) decât numărul de puncte câștigate de fiecare din jucătorii echipei F (1, 2, 3). A treia probă s-a încheiat cu remiză deoarece niciun jucător al niciunei echipe nu obține un număr de puncte strict mai mare decât toate punctajele obținute de jucătorii celeilalte echipe.

	ktlon.in	ktlon.out	**Explicații**
1)			A patra probă s-a încheiat tot cu remiză, deoarece toți jucătorii au obținut exact aceleași punctaje ca și la proba a treia. Răspunsul este 1 deoarece echipa R a câștigat o singură probă.
2)	2 3 3 8 8 5 7 7 7 1 2 3 3 5 3 4 1 2 6 5 1	7	Se rezolvă cerința **2**. Echipa F câștigă prima probă și primește 2 stele (M=2, (8+8)-(7+7)=2). Echipa R câștigă a doua probă și primește 2 stele (M=1, 5-3=2). Echipa R câștigă a treia probă și primește 5 stele (M=2, (6+5)-(4+2)=5). În total, echipa F a primit 2 stele iar echipa R a primit 7 stele. Competiția este câștigată de echipa R cu 7 stele.

Descrierea soluției

Pentru rezolvarea cerinței 1 se determină pentru fiecare din cele k probe cel mai mare punctaj obținut de jucătorii echipei F și apoi cel mai mare punctaj obținut de jucătorii echipei R. Dacă punctajul maxim corespunzător echipei R este strict mai mare decât punctajul maxim al echipei R se incrementează numărul de probe câștigate de echipa R.

Pentru rezolvarea cerinței 2, o abordare posibilă este următoarea: pentru fiecare din cele k probe, atât pentru echipa F, cât și pentru echipa R, se determină cele mai mari cinci punctaje obținute de jucătorii echipei, dacă $n \geq 5$, sau primele n punctaje în caz contrar, în ordine descrescătoare. Cât timp primele cel mult cinci punctaje maxime corespunzătoare unei echipei sunt strict mai mari decât punctajul maxim obținut de jucătorii celeilalte echipe se adună la numărul de stele corespunzător acesteia numărul de stele obținut la proba curentă, conform enunțului problemei. Se afișează la final numărul maxim de stele obținut de o echipă.

1.2 Problema iepuraș

Propunător: prof Gina Balacea
Colegiul Național "Vasile Alecsandri" Galați

Enunț

Pentru că îi plac cifrele, Skippie, iepurașul norocos, a stabilit cum se obține cifra de control a unui număr: se efecturază suma cifrelor sale, apoi suma cifrelor acestei sume, până când suma obținută este un număr format dintr-o singură cifră. Această ultimă cifră, spune Skippie, poartă numele de cifră de control.

Skippie, a ascuns în pădure n ouă roșii. Pe fiecare ou a pictat câte un număr natural nenul. Iar acum se întreabă care este suma dintre cel mai mare și cel mai mic număr natural care se pot forma din toate cifrele distincte folosite în scrierea numărului pictat.

În plus, pentru că lui Skippie îi plac problemele complicate, pentru fiecare număr pictat pe câte un ou, el ar vrea să afle și de câte ori apare cifra de control a numărului în scrierea tuturor numerelor naturale mai mici sau egale decât numărul pictat.

Cerințe

1. Pentru fiecare dintre cele n numere pictate de Skippie, aflați suma dintre cel mai mare și cel mai mic număr natural care se pot forma din toate cifrele distincte folosite în scrierea numărului pictat.
2. Pentru fiecare dintre cele n numere pictate de Skippie, aflați de câte ori apare cifra de control a numărului pictat în scrierea tuturor numerelor naturale mai mici sau egale decât numărul pictat.

Date de intrare

Fișierul de intrare *iepuras.in* conține un număr natural C. Acesta poate avea valorile 1 sau 2 și reprezintă cerința problemei. Cea de-a doua linie a fișierului de intrare conține un număr natural n reprezentând numărul de ouă roșii pictate de Skippie. Fiecare dintre următoarele n linii ale fișierului de intrare conține câte un număr natural nenul reprezentând numerele pictate de iepuraș pe cele n ouă roșii.

Date de ieșire Fișierul de ieșire iepuras.out va conține n numere întregi, fiecare pe o linie separată. În ordinea apariției numerelor pictate de iepuraș în fișierul de intrare, se afișează răspunsurile la cerința C.

Restricții și precizări

Pentru teste în valoare de	Cerința este	Numărul de ouă este	Numerele pictate de iepuraș sunt mai mici sau egale cu
16 puncte	$C = 1$	$n = 1$	10^9
24 de puncte	$C = 1$	$1 < n \leq 100\ 000$	10^9
24 de puncte	$C = 2$	$1 \leq n \leq 100$	2 000
36 de puncte	$C = 2$	$100 < n \leq 100\ 000$	10^{18}

Exemple

	iepuras.in	iepuras.out	Explicații
1)	1 2 121 33343	33 77	Se rezolvă cerința **1**. Sunt 2 ouă pictate ($n = 2$). Pentru primul ou, pictat cu numărul 121: - cel mai mare număr natural cu cifre distincte format cu toate cifrele distincte ale numărului pictat este 21; - cel mai mic număr natural cu cifre distincte format cu toate cifrele distincte ale numărului pictat este 12. Deci suma celor două numere este 33($21 + 12 = 33$). Pentru al doilea ou, pictat cu numărul 33343: - cel mai mare număr natural cu cifre distincte format cu toate cifrele distincte ale numărului pictat este 43; - cel mai mic număr natural cu cifre distincte format cu toate cifrele distincte ale numărului pictat este 34. Deci suma celor două numere este 77 ($43 + 34 = 77$).
2)	2 2 123 191	22 39	Se rezolvă cerința **2**. Sunt 2 ouă pictate ($n = 2$). Pe primul ou este scris numărul 123 iar pe al doilea ou numărul 191. Cifra de control a numărului 123 este 6 ($1 + 2 + 3 = 6$). Numărul de apariții a cifrei 6 în scrierea a tuturor numerelor naturale mai mici sau egale cu 123 este 22. Cifra 6 apare în scrierea numerelor: 6, 16, 26, 36, 46, 56, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 76, 86, 96, 106, 116 de 22 ori. Cifra de control a numărului 191 este 2. ($1 + 9 + 1 = 11$; $1 + 1 = 2$). Numărul de apariții a cifrei 2 în scrierea a tuturor numerelor naturale mai mici sau egale cu 191 este 39. Cifra 2 apare în scrierea numerelor **2**, 1**2**, **2**0, **2**1, **22**, **2**3, **2**4, **2**5, **2**6,**27**, **2**8, **2**9, 3**2**, 4**2**, 5**2**, 6**2**, 7**2**, 8**2**, 9**2**, 10**2**, 11**2**, 1**2**0, 1**2**1, 1**22**, 1**2**3, 1**2**4, 1**2**5, 1**2**6, 1**27**, 1**2**8, 1**2**9, 1**3**2, 1**4**2, 1**5**2, 1**6**2, 1**7**2, 1**8**2 de 39 de ori.

Descrierea soluției

Pentru rezolvarea problemei se utilizează tipuri de date simple: întregi și un vector de frecvență. În plus, se utilizează elemente de matematică din programa școlară a clasei a V-a. (Operații cu numere naturale, scrierea în baza 10 a unui număr natural) și structuri elementare de control (structura alternativă și structura repetitivă).

Cerința 1

Se descompune în cifre fiecare număr N dintre cele n numere citite și se reține numărul de apariții al cifrelor lui N într-un vector $cif[10]$ (cu cel mult 10 elemente).

Pentru determinarea numărului maxim (N_{max}) care se poate obține din cifrele distincte ale numărului N se parcurge vectorul de frecvență, de la poziția 9 către poziția 0. Dacă cifra i apare în scrierea zecimală a numărului N ($cif[i] \neq 0$) atunci $N_{max} = N_{max} \times 10 + i$.

Pentru determinarea numărului minim (N_{min}) care se poate obține din cifrele distincte ale numărului N se parcurge vectorul de frecvență, de la poziția 1 către poziția 9 și se caută prima valoare nenulă. Fie aceasta j. Atunci $N_{min} = j$ și se marchează faptul că s-a folosit cifra j, atribuind lui $cif[j]$ valoarea 0. Se parcurge apoi vectorul de frecvență de la 0 către 9. Dacă cifra i apare în scrierea zecimală a numărului N ($cif[i] \neq 0$) atunci $N_{min} = N_{min} \times 10 + i$.

Se afișează suma dintre cele două numere obținute, N_{max} și N_{min}.

Pentru fiecare număr N, dintre cele n numere citite, se inițializează vectorul de frecvență $cif[]$ cu 0.

Cerința 2

Pentru de a putea rezolva cerința a doua, mai întâi vom analiza câteva cazuri particulare și vom încerca să răspundem la câteva întrebări:

(1) Câte numere mai mici ca 2**4**3 au cifra **5** pe poziția zecilor?

50	51	52	53	54	55	56	57	58	59
150	151	152	153	154	155	156	157	158	159

Răspuns: 20

(2) Câte numere mai mici ca 2**5**3 au cifra **5** pe poziția zecilor?

50	51	52	53	54	55	56	57	58	59
150	151	152	153	154	155	156	157	158	159
250	251	252	253						

Răspuns: 24

(3) Câte numere mai mici ca 2**6**3 au cifra **5** pe poziția zecilor?

50	51	52	53	54	55	56	57	58	59
150	151	152	153	154	155	156	157	158	159
250	251	252	253	254	255	256	257	258	259

Răspuns: 30

Analizând cu atenție aceste trei exemple, ne dăm seama că se disting trei cazuri. Pentru fiecare dintre aceste cazuri putem deduce câte o regulă generală prin care să calculăm răspunsurile:

Fie: s = cifra sutelor, z = cifra zecilor, u = cifra unităților

Atunci:

- dacă $z < 5$, atunci răspunsul este $s \times 10$
- dacă $z = 5$, atunci răspunsul este $s \times 10 + u + 1$
- dacă $z > 5$, atunci răspunsul este $(s + 1) \times 10$

Dacă vom analiza exemple cu mai multe cifre, putem generaliza următoarele reguli:

Fie un număr care scris pe cifre arată astfel: $ss \ldots ss\mathbf{z}uu \ldots uu$.

Fie CC cifra sa de control, iar nrU numărul de cifre notate cu u în scrierea de mai sus.

Atunci:

- dacă $z < CC$, atunci numărul de numere mai mici sau egale cu $ss \ldots ss\mathbf{z}uu \ldots uu$ care conțin cifra CC pe poziția cifrei notată cu z este $ss..ss * 10^{nrU}$
- dacă $z = CC$, atunci numărul de numere mai mici sau egale cu $ss \ldots ss\mathbf{z}uu \ldots uu$ care conțin cifra CC pe poziția cifrei notată cu z este $ss..ss * 10^{nrU} + uu..uu + 1$
- dacă $z > CC$, atunci numărul de numere mai mici sau egale cu $ss \ldots ss\mathbf{z}uu \ldots uu$ care conțin cifra CC pe poziția cifrei notată cu z este $(ss..ss + 1) * 10^{nrU}$

Calculând în prealabil cifra de control, iterând prin cifrele numărului și cu ajutorul acestor reguli, putem calcula răspunsul cerinței a doua.

Cerința 2 (explicație alternativă)

Cifra de control a unui număr natural nenul este egală cu 9 dacă numărul este multiplu de 9 sau este egală cu restul împărțirii la 9 a numărului dat, dacă restul este nenul.

Soluția problemei se bazează pe scrierea în baza 10 a unui număr natural.

Pentru fiecare dintre cele n numere scrise de iepuraș pe ouă se determină cifra de control.

Numărul de apariții al cifrei de control în scrierea tuturor numerelor naturale nenule mai mici sau egale cu un număr natural N se determină analizând scrierea în baza 10 a numărului.

Fie n_0 cifra unităților, n_1 este cifra zecilor, ... și n_k cifra cea mai semnificativă a numărului natural N. Atunci $N = n_0 + n_1 \times 10 + n_2 \times 10^2 + \ldots + n_k \times 10^k$.

Notăm cu c cifra de control a numărului natural nenul N.

Soluția care analizează apariția cifrei de control c în scrierea tuturor numerelor naturale nenule mai mici sau egale cu numărul N descompunând în cifre toate numerele din interval rezolvă suficient de rapid grupa de teste în valoare de 24 de puncte.

Pentru punctaj maxim însă, va trebui să găsim un alt mod de a calcula răspunsul.

Se calculează de câte ori apare cifra c pe poziția unităților în toate numerele naturale mai mici sau egale cu N. Astfel, cifra c apare pe poziția unităților de $N/10$ ori. Dacă n_0 (cifra unităților numărului N) este mai mare sau egal cu c atunci numărul de apariții al cifrei de control se mărește cu 1.

Se calculează apoi de câte ori apare cifra c pe poziția zecilor în toate numerele naturale mai mici sau egale cu N. Astfel, cifra c apare pe poziția zecilor de $N/100$ ori. Dacă n_1 (cifra zecilor numărului N) este mai mare decât c atunci numărul de apariții al cifrei de control se mărește cu 10, în schimb, dacă n_1 este egală cu cifra de control atunci numărul de apariții al cifrei de control se mărește cu $n_0 + 1$.

În general, se calculează de câte ori apare cifra c pe poziția p în toate numerele naturale mai mici sau egale cu N. Astfel, cifra c apare pe poziția p de $N/10^{(p+1)}$ ori. Dacă n_p este mai mare decât c atunci numărul de apariții al cifrei de control se mărește cu 10^p, în schimb, dacă n_p este egală cu c atunci numărul de apariții al cifrei de control se mărește cu $(N\%10^p) + 1$.

De exemplu, pentru numărul 123, cifra de control este 6.

Cifra 6 apare pe poziția unităților de 12 ori, pe poziția zecilor de $1 \times 10 = 10$ ori și niciodată pe poziția sutelor. În total, în scrierea zecimală a numerelor $1, 2, 3, 4, 5, 6, 7, \ldots, 60, 61, 62, \ldots, 123$ cifra 6 apare de 22 de ori.

1.3 Problema taieri

Propunător: prof Marius Nicoli
Colegiul Național "Frații Buzești" Craiova

Enunț

Avem la dispoziție n bare metalice cu aceeași grosime, dar lungimi diferite. Putem alege oricare bară și să o tăiem, obținând alte două bare, de lungimi mai mici. Ne dorim ca, folosind doar această operație (deci fără să le putem suda), să obținem un număr de bare de anumite lungimi date. Mai exact, dându-se un set de 4 numere a, b, c, d, trebuie să decidem dacă putem obține a bare de lungime 1, b bare de lungime 2, c bare de lungime 4 și d bare de lungime 8. Odată aplicată o tăiere de lungime L asupra unei bare, restul poate fi în continuare folosit pentru a tăia alte bare de oricare dintre lungimile dorite.

Cerință

Cunoscând n - numărul de bare metalice și lungimile celor n bare metalice avute la dispoziție, pentru fiecare din seturile de 4 numere a b c d date, determinați dacă, pornind de la cele n lungimi date, se pot obține barele de lungimile dorite.

Date de intrare

Pe prima linie a fișierului de intrare *taieri.in* se află numărul natural n. Pe linia a doua se afla cele n numere naturale ce reprezintă lungimile barelor inițiale. Pe linia a treia se află un număr natural m ce reprezintă numărul de seturi de 4 numere. Pe fiecare din următoarele m linii se află câte patru numere naturale a b c d cu semnificația de mai sus.

Numerele de pe aceeași linie sunt separate prin câte un spațiu.

Date de ieșire

Fișierul de ieșire *taieri.out* va conține m valori 0 și 1, separate prin câte un spațiu. Pentru fiecare set de 4 numere, în ordinea apariției lor în fișierul de intrare, se scrie în fișierul de ieșire valoarea 1, dacă se poate obține numărul dorit de bare pentru toate cele 4 lungimi din setul corespunzător, respectiv 0 în caz contrar.

Restricții și precizări

- Lungimile barelor sunt numere naturale nenule cel mult egale cu 10 000 000.
- $0 \leq a, b, c, d \leq 10\ 000\ 000$.
- Pentru teste în valoare de 16 puncte $1 \leq n \leq 1\ 000$, $1 \leq m \leq 1\ 000$, iar pentru toate seturile dintr-un test avem $b = 0$, $c = 0$ și $d = 0$.
- Pentru alte teste în valoare de 28 puncte $1 \leq n \leq 1\ 000$ și $1 \leq m \leq 1\ 000$.
- Pentru alte teste în valoare de 56 de puncte $1 \leq n \leq 100\ 000$ și $1 \leq m \leq 100\ 000$.

Exemplu

taieri.in	taieri.out	Explicații
5 10 12 8 3 1 3 2 3 2 2 31 0 0 0 1 13 0 1	1 1 0	**La primul set** - 2 3 2 2 trebuie obținute două bare de lungime 1, trei bare de lungime 2, două de lungime 4 și două de lungime 8. Putem tăia bara de lungime 10 în una de 8 și una de 2. Avem deja a doua bară de lungime 8. Ne mai rămân astfel bare cu lungimile: 2, 12, 3 și 1. Cele două bare de lungime 4 le putem tăia din cea de lungime 12, rămânându-ne bare de lungimile 2, 4, 3 și 1. Pentru cele 3 de lungime 2 putem folosi prima bară și pe a doua o tăiem în două bucăți, iar pe cele 4 de lungime 1 le putem obține folosind barele cu lungimile 3 și 1 rămase. **La al doilea set** - 31 0 0 0 putem obține din prima bară dată 10 bare de lungime 1, din a doua 12 bare de lungime 1, din a treia încă 8 bare de lungime 1 și mai avem ultima bară deja de lungime 1, așadar se pot obține cele 31 de bare necesare. Remarcați că nu a fost nevoie să folosim bara de lungime 3. Se observă că nu este nevoie să obținem bare de lungimile 2, 4, 8. **La al treilea set** - 1 13 0 1, oricum am tăia barele avute la dispoziție, nu putem obține întreg setul format din: o bară de lungime 1, 13 bare de lungime 2 și o bară de lungime 8.

Descrierea soluției

Pentru un set a, b, c, d dat la intrare putem parcurge șirul pentru a determina mai intai dacă se pot obține d bare de lungime 8. De exemplu, dacă o bară dată are lungimea $V[i]$, din ea putem obține $V[i]/8$ bare de lungime 8. Înlocuim $V[i]$ cu valoarea rămasă pentru a putea utiliza acum $V[i]$ și în scopul tăierii unor bare de lungimi mai mici.

Dacă reușim să obținem din tot șirul d bare de lungime 8, putem relua procedeul pentru bare de lungimi 4, apoi 2 și apoi 1.

Aplicănd metoda descrisă mai sus pentru fiecare set din fișierul de intrare obținem un algoritm corect dar care nu se încadrează în timp pe testele mari (la fiecare set de la intrare ar fi necesară traversarea întregului șir de valori inițiale).

Următoarea observație ne permite să obținem o soluție mai rapidă.

Dacă putem obține bare de o anumită lungime L ca surplus față de ce este necesar, se poate să le tăiem pe jumătate și le luăm în calcul apoi pentru lungimea $L/2$.

Astfel, fără să ne intereseze pentru început valorile din seturile de câte 4 numere, putem determina pentru șirul celor n lungimi următoarele valori:

- Numărul maxim de bare de lungime 8 (notat $Nr[8]$) care se pot obține (adunăm la o variabilă valorila $V[i]/8$ pentru toate elementele din vector); Înlocuim apoi fiecare valoare cu $V[i]\%8$.
- Pe noul vector, reluăm procedeul pentru lungimile $4, 2, 1$, calcuând astfel 4 valori.

Apoi, pentru un set de 4 lungimi date a, b, c, d este suficient sa consultăm cele 4 valori calculate anterior ($Nr[1]$, $Nr[2]$, $Nr[4]$, $Nr[8]$). Dacă $d > Nr[8]$, nu avem soluție. În caz contrar adunăm

la $Nr[4]$ valoarea $(Nr[8] - d) \times 2$ (surplusul de lungime 8 permite obținerea de număr dublu de bare de lungime 4). Același raționament îl aplicăm apoi pentru 4, 2, 1, în această ordine.

Această soluție permite traversarea vectorului initial de un număr constant de valori pentru a obține informațiile Nr, iar în etapa a doua, pentru fiecare set a, b, c, d este suficient să interogăm doar cele 4 valori Nr.

Capitolul 2

Clasa a VI-a

2.1 Problema Butoi

Propunător: prof Marinel Șerban

Colegiul Național "Emil Racoviță" Iași

Enunț

Vară, căldură mare. Gigel se joacă în curte udând florile. După ce a terminat, mama lui îi dă o sarcină mai grea. Gigel trebuie să umple un butoi cu apă de rezervă în caz de secetă. Dar nu oricum! El are la dispoziție un șir de găleți de diferite capacități și trebuie să le folosească doar pe acestea pentru umplerea completă a butoiului. O *operație* constă în umplerea completă a unei o găleți de la sursa de apă și golirea ei în butoi. Desigur, o găleată se poate folosi de mai multe ori. Butoiul are capacitate de V litri, Gigel are N găleți de capacități c_1, c_2,...,c_N litri, numere întregi și distincte și poate folosi o găleată de cel mult K ori. Ajutați-l pe Gigel să umple butoiul.

Cerințe

Scrieţi un program care să rezolve următoarele cerințe:

1. Determinați câte modalități de umplere a butoiului există;
2. Determinați o modalitate de umplere a butoiului cu număr minim de operații;
3. O secvență de exact P găleți alăturate se numește *norocoasă* dacă prin efectuare operației de același număr de ori pentru fiecare dintre ele, vom putea umple complet butoiul. Determinați secvența *norocoasă* care permite folosirea celor P găleți de un număr minim de ori pentru umplerea completă a butoiului. Secvența norocoasă se va identifica prin poziția primei găleți folosite. Dacă există mai multe soluții se va afișa cea cu poziția minimă. Găletile din secvența norocoasă **se pot folosi de câte ori este nevoie**.

Date de intrare

Fişierul de intrare **butoi.in** conţine pe prima linie, un număr natural C care poate avea valorile 1, 2 sau 3 şi reprezintă numărul cerinţei. Linia a doua conține patru valori naturale V N K P, separate prin câte un spaţiu, reprezentând în ordine capacitatea butoiului, numărul de găleți, numărul maxim de operații care poate fi efectuat cu o găleată, în cazul cerințelor 1 și 2, iar ultimul număr reprezintă lungimea secvenței *norocoase* căutate la cerința 3. Linia a treia

conţine N valori naturale distincte $c_1, c_2, ..., c_N$, separate prin câte un spațiu, reprezentând, în ordine, capacitățile celor N găleți din șir.

Date de ieșire

Fişierul de ieşire **butoi.out** conţine pentru cerința 1, pe prima linie, un întreg reprezentând numărul de modalități de a umple butoiul. Pentru cerința 2 prima linie va conține N valori naturale separate prin câte un spațiu, reprezentând numărul de utilizări a fiecărei găleți iar pentru cerința 3 prima linie va conține un număr natural reprezentând poziția din șir a primei găleți din secvența norocoasă determinată.

Restricții și precizări

- $5 \leq V \leq 360\ 000$
- Pentru cerințele 1 și 2 restricțiile sunt: $1 \leq N \leq 9$; $1 \leq c_i \leq 8\ 000$ și $1 \leq K \leq 5$
- Pentru cerința 3 restricțiile sunt: $3 \leq N \leq 100\ 000$; $1 \leq c_i \leq 200\ 000$; $3 \leq P \leq 10\ 000$ și $P \leq N$
- Pentru cerința 3 capacitățile c_1, c_2,...,c_N ale găleților nu sunt neapărat distincte
- Pentru rezolvarea corectă a primei cerințe se obțin 30 de puncte, pentru rezolvarea corectă a celei de a doua cerințe se obțin 40 de puncte, iar pentru rezolvarea corectă a celei de a treia cerințe se obțin 30 de puncte.

Exemple

1)	butoi.in	butoi.out	**Explicații**
	1 **90 7 1 2** **30 56 70 34 60 15 5**	3	Se rezolvă cerința **1**. Există 3 modalități de a umple butoiul folosind **cel mult o dată** fiecare găleată, deoarece $K = 1$. Acestea sunt 30 + 60, 56 + 34 și 70 + 15 + 5.
2)	butoi.in	butoi.out	**Explicații**
	1 **2020 6 5 3** **17 42 200 101 51 171**	19	Se rezolvă cerința **1**. Există 19 modalități de a umple butoiul.
3)	butoi.in	butoi.out	**Explicații**
	2 **2020 6 5 3** **17 42 200 101 51 171**	0 0 5 3 4 3	Se rezolvă cerința **2**. Numărul minim de operații este 15. Găletile au fost folosite astfel: $0 * 17 + 0 * 42 + 5 * 200 + 3 * 101 + 4 * 51 + 3 * 171 =$ $= 0 + 0 + 1000 + 303 + 204 + 513 = 2020$.
4)	butoi.in	butoi.out	**Explicații**
	3 **120 7 1 3** **90 12 20 8 41 80 11**	2	Se rezolvă cerința **3**. Secvența norocoasă de lungime 3 determinată începe la poziția 2. Prin folosirea de 3 ori a fiecărei găleți din secvență butoiul se umple.

Descrierea soluției

Pentru cerința1 și cerința2

Notăm cu $t_1, t_2..t_n$ numărul de utilizări pentru fiecare tip de găleată.

Se folosește un **algoritm de tip succesor** prin adunare cu 1 în baza $k+1$ pe n poziții. Algoritmul începe de la combinația nulă:

$$t_1 = 0; t_2 = 0; \ldots; t_{n-1} = 0; t_n = 0 \tag{2.1}$$

După o adunare cu 1 o să obținem:

$$t_1 = 0; t_2 = 0; \ldots; t_{n-1} = 0; t_n = 1 \tag{2.2}$$

După k adunări cu 1 o să obținem:

$$t_1 = 0; t_2 = 0; \ldots; t_{n-1} = 0; t_n = k \tag{2.3}$$

Iar după $k+1$ adunări de 1:

$$t_1 = 0; t_2 = 0; \ldots; t_{n-2} = 0; t_{n-1} = 1; t_n = 0 \tag{2.4}$$

Cantitatea totală de apă se calculează folosind formula:

$$V = t_1 * c_1 + t_2 * c_2 + ... + t_n * c_n \tag{2.5}$$

După fiecare dintre cele k^n adunări cu 1 trebuie să verificăm care este volumul total (folosim formula (2.5)).

Pentru cerința 1 se numără încă o soluție și se trece la combinația următoare.

Pentru cerința 2 se verifică dacă combinația are mai puține operații și se reține minimul apoi se trece la combinația următoare.

Cantitatea totală de apă se calculează folosind formula: $V = t_1 * c_1 + t_2 * c_2 + ... + t_n * c_n$.

Soluții alternative

- Simularea tuturor combinațiilor posibile folosind $1, 2..9$ structuri repetitive incluse una în alta.
- Soluție de tip backtracking (care nu se încadrează în materia clasei a 6-a)

Pentru cerința 3

Se calculează suma S a oricăror p capacități succesive. Dacă V se divide la S atunci am detectat o soluție posibilă. Dintre toate soluțiile găsite o reținem și afișăm pe cea care folosește numărul minim de găleți.

Pentru calcularea lui S se pot folosi următoarele metode:

- Se fixează capătul stânga al unui interval de lungime p și se face suma elementelor. (scor obținut 66% puncte pe cerința 3)
- Se calculează suma primelor p elemente, iar apoi se mișcă intervalul câte un singur pas la dreapta. Pentru un intervalul $[st, dr]$ se preia S-ul de la intervalul precedent $[st-1, dr-1]$, se scade elementul eliminat $c[st-1]$ și se adaugă elementul nou introdus $c[dr]$ (scor obținut 100% puncte pe cerința 3)
- Se folosesc sume parțiale pe prefixe. Notăm cu $partial[i] = c[1] + c[2] + .. + c[i]$. Suma elementelor pe intervalul $[st, dr] = partial[dr] - partial[st-1]$ (scor obținut 100% puncte pe cerința 3)

Observații

Problema testează cunoștințe despre:

- lucrul cu tablouri unidimensionale
- adunare pe vector într-o bază de numerație
- lucrul cu baze de numerație
- determinare minim
- tehnica algoritm de tip succesor
- sume parțiale pe vector

2.2 Problema Păsări

Propunător: prof. Daniela Lica
Centrul Județean de Excelență Prahova, Ploiești

Enunț

În grădina lui Macarie, există N^2 pomi fructiferi, de diferite înălțimi, plantați sub forma unui caroiaj de N linii și N coloane, numerotate de la 1 la N. Cum există păsări care mănâncă fructele lor, Macarie s-a gândit să plaseze sisteme de supraveghere, care asigură protecție într-o oarecare măsură a pomilor situați pe linia și coloana unde a fost amplasat, în toate cele 4 sensuri de deplasare Nord, Sud, Est și Vest, adică la stânga și la dreapta în cadrul liniei și în sus și în jos în cadrul coloanei. În oricare din aceste patru sensuri de deplasare, supravegherea asigurată de un sistem se întrerupe când întâlnește un pom de înălțime strict mai mare decât a pomului în care a fost amplasat.

Cerințe

Cunoscând înălțimile tuturor celor N^2 pomi, scrieți un program care:

1. Presupunând că Macarie are un singur sistem de supraveghere determinați pentru o linie dată L, care este coloana pomului în care trebuie amplasat sistemul de supraveghere astfel încât să fie protejați un număr maxim de pomi. Dacă există mai multe soluții, se va afișa coloana minimă.

2. Presupunând că Macarie a amplasat în fiecare pom al grădinii sale, câte un sistem de supraveghere, determinați linia și coloana pomului protejat de cele mai multe sisteme de supraveghere. Dacă există mai multe soluții se va afișa soluția cu linia minimă, iar în cazul în care liniile sunt egale, cea cu coloana minimă.

Date de intrare

Fişierul de intrare **pasari.in** conţine pe prima linie, un număr natural C care poate avea valorile 1, sau 2 şi reprezintă numărul cerinţei care trebuie rezolvată. Dacă $C = 1$ atunci pe a doua linie se află perechea de numere naturale N și L separate printr-un spațiu, iar dacă $C = 2$, pe cea de-a doua linie se află un număr natural N având semnificația din enunț. Pe următoarele N linii, câte numere N naturale despărțite prin câte un spațiu, reprezentând în ordine, înălțimile pomilor de pe fiecare rând, începând cu primul până la ultimul.

Date de ieșire

Fişierul de ieşire **pasari.out** va conţine pe prima linie:

- dacă $C = 1$ un singur număr natural, reprezentând coloana minimă determinată conform cerinței
- dacă $C = 2$ două numere naturale, reprezentând linia și coloana determinată conform cerinței

Restricții și precizări

- $5 \leq N \leq 1\ 000$
- 1 ≤ înălțimea oricărui pom ≤ 100 000
- Pentru rezolvarea corectă a primei cerințe se obțin 30 de puncte, iar pentru rezolvarea corectă a celei de a doua cerințe se obțin 70 de puncte.

Exemple

1)	pasari.in	pasari.out	**Explicații**
	1 5 2 8 4 3 9 1 5 1 4 2 1 6 7 2 3 1 7 6 4 1 1 1 1 1 1 1	3	Se rezolvă cerința **1**. Pe linia 2 așezarea optimă este pe coloana 3, în felul acesta fiind protejați 8 pomi. În exemplu pomii protejați de sistemul plasat pe coloana a treia sunt marcați cu roșu.
2)	**pasari.in**	**pasari.out**	**Explicații**
	2 5 1 2 3 2 1 2 2 3 2 2 3 2 3 2 3 2 2 3 2 2 1 2 3 2 1	3 2	Se rezolvă cerința **3**. Pomul situat la pozițiile (3,2), adică pe linia 3 și coloana 2, este supravegheat de opt sisteme. Acestea sunt situate la pozițiile (1,2), (2,2), (3,2), (4,2), (5,2), (3,1), (3,3) și (3,5) și sunt marcate în exemplu cu albastru. Pomul situat la pozițiile (3,4) este protejat tot de 8 sisteme de supraveghere, dar conform regulii din enunț se va afișa cel situat pe linia mai mică, respectiv 3 2.

Descrierea soluției

Notații:

- $H[i][j]$ $(1 \leq i \leq N, 1 \leq j \leq N)$ = înălțimea pomului amplasat pe linia i și coloana j. Matricea $H[][]$ este citită din fișierul de intrare;
- $Nord[i][j] = x$ $(1 \leq i \leq N, 1 \leq j \leq N, x < i) \Leftrightarrow x$ este indicele maxim al unei linii, astfel încât $H[x][j] > H[i][j]$; dacă nu există niciun astfel de x, se consideră că $Nord[i][j] = 0$. Mai informal spus, x reprezintă indicele liniei pomului de pe coloana j care va întrerupe supravegherea, pe direcția $Nord$, generată de un sistem situat în pomul de pe linia i și coloana j (adică, primul pom strict mai înalt decât acesta, în sus);
- $Sud[i][j] = x$ $(1 \leq i \leq N, 1 \leq j \leq N, x > i) \Leftrightarrow x$ este indicele minim al unei linii, astfel încât $H[x][j] > H[i][j]$; dacă nu există niciun astfel de x, se consideră că $Sud[i][j] = N+1$. Mai informal spus, x reprezintă indicele liniei pomului de pe coloana j care vă întrerupe supravegherea, pe direcția Sud, generată de un sistem situat în pomul de pe linia i și coloana j (adică, primul pom strict mai înalt decât acesta, în jos);
- $Vest[i][j] = y$ $(1 \leq i \leq N, 1 \leq j \leq N, y < j) \Leftrightarrow y$ este indicele maxim al unei coloane, astfel încât $H[i][y] > H[i][j]$; dacă nu există niciun astfel de y, se consideră că $Vest[i][j] = 0$. Mai informal spus, y reprezintă indicele liniei pomului de pe linia i care va întrerupe supravegherea, pe direcția $Vest$, generată de un sistem situat în pomul de pe linia i și coloana j (adică, primul pom strict mai înalt decât acesta, la stânga);

- $Est[i][j] = y$ $(1 \leq i \leq N, 1 \leq j \leq N, y > j) \Leftrightarrow y$ este indicele minim al unei coloane, astfel încât $H[i][y] > H[i][j]$; dacă nu există niciun astfel de y, se consideră că $Est[i][j] = N+1$. Mai informal spus, y reprezintă indicele liniei pomului de pe linia i care va întrerupe supravegherea, pe direcția Est, generată de un sistem situat în pomul de pe linia i și coloana j (adică, primul pom strict mai înalt decât acesta, la dreapta);

Cerința 1 (30 de puncte)

- În cadrul acestei cerințe, avem la dispoziție un singur dispozitiv de supraveghere, pe care dorim să îl amplasăm într-un pom situat pe linia L, astfel încât numărul pomilor supravegheați de acesta să fie maxim.
- Putem spune că un dispozitiv amplasat în pomul de pe linia i și coloana j $(1 \leq i \leq N$, $1 \leq j \leq N)$ va supraveghea un număr de pomi (inclusiv pomul de pe linia i și coloana j) egal cu $Extindere[i][j]$, după cum arată formula următoare:

$$Extindere[i][j] = (Sud[i][j] - Nord[i][j] - 1) + (Est[i][j] - Vest[i][j] - 1) - 1$$

- Prin urmare, pentru a rezolva corect cerința, trebuie să identificăm cel mai mic indice al unei coloane j, pentru care $Extindere[L][j]$ este maxim, după cum urmează:

```
int ans = 0, Max = 0;

for(int j = 1; j <= N; ++j)
    if(Extindere[L][j] > Max)
        Max = Extindere[L][j], ans = j;

afișează ans;
```

- Având în vedere că sistemul de supraveghere trebuie amplasat pe linia L, primită în cadrul datelor de intrare, înseamnă că ne vor interesa doar valorile de pe linia L din matricele $Nord[][]$, $Sud[][]$, $Vest[][]$, $Est[][]$, și implicit $Extindere[][]$.
- Valorile $Nord[L][j]$, $Sud[L][j]$, $Vest[L][j]$, $Est[L][j]$ $(1 \leq j \leq N), j - fixat$ se pot calcula în complexitatea de timp $O(N)$, utilizând o structură repetitivă, iterând un indice până când am întâlnit un pom mai înalt decât cel curent (linia L, coloana j), sau am ieșit din matrice.
- Pentru a calcula $Nord[L][j]$ și $Sud[L][j]$ pentru un $j - fixat$ $(1 \leq j \leq N)$, putem proceda astfel:

```
Nord[L][j] = 0, Sud[L][j] = N + 1;

for(int x = L - 1; x >= 1; --x)
    if(H[x][j] > H[L][j])
    {
        Nord[L][j] = x;

        break;
    }

for(int x = L + 1; x <= N; ++x)
    if(H[x][j] > H[L][j])
    {
        Sud[L][j] = x;

        break;
    }
```

- În mod similar, se pot calcula valorile $Vest[L][j]$ și $Est[L][j]$ pentru un $j - fixat$ $(1 \leq j \leq N)$.
- O soluție care are complexitatea totală de timp $O(N \cdot N)$ obține punctajul maxim pentru $C = 1$.

Cerința 2 (70 de puncte)

- În cadrul acestei cerințe, se știe faptul că în fiecare pom dintre cei $N \cdot N$ este amplasat câte un dispozitiv de supraveghere. Astfel, pentru fiecare linie i și coloană j $(1 \leq i \leq N$, $1 \leq j \leq N)$ avem că: $Extindere[i][j] \geq 1$ (dispozitivul va supraveghea, cu siguranță, cel puțin un pom, și anume pomul de pe linia i și coloana j).
- Se cere să se afle care sunt *coordonatele* (adică linia și coloana) pomului care este supravegheat de cele mai multe dispozitive.
- Dispozitivul de supraveghere amplasat în pomul de pe linia i și coloana j $(1 \leq i \leq N$, $1 \leq j \leq N)$ va supraveghea pomii:
  - pe ***verticală*** (direcția $Nord$ - Sud):
    - $(Nord[i][j] + 1, j)$;
    - $(Nord[i][j] + 2, j)$;
    - ...
    - $(i - 1, j)$;
    - $(i + 1, j)$;
    - ...
    - $(Sud[i][j] - 2, j)$;
    - $(Sud[i][j] - 1, j)$.
  - pe ***orizontală*** (direcția $Vest$ - Est):
    (i, Vest[i][j] + 1); (i, Vest[i][j] + 2); ...; (i, j - 1); (i, j + 1); ...;
    (i, Est[i][j] - 2); (i, Est[i][j] - 1)
  - în care este amplasat: (i, j).
- Așadar, pentru fiecare pom, vom introduce notația: $cnt[i][j] = V \Leftrightarrow$ pomul situat pe linia i și coloana j va fi supravegheat de V dispozitive $(1 \leq i \leq N, 1 \leq j \leq N)$. Răspunsul va fi dat de indicii liniei x, respectiv coloanei y pentru care valoarea $cnt[x][y]$ este maximă, după cum urmează:

```
int ans_i = 0, ans_j = 0, Max = 0;

for(int x = 1; x <= N; ++x)
    for(int y = 1; y <= N; ++y)
        if(cnt[x][y] > Max)
            Max = cnt[x][y], ans_i = x, ans_j = y;

afișează ans_i ans_j;
```

- O soluție care are complexitatea de timp $O(N^3)$, asemănătoare cu cea de la cerința 1, prin care se utilizează $O(N)$ operații pentru fiecare dintre cele $N * N$ dispozitive, și marchează corespunzător modificările în matricea $cnt[][]$, nu va reuși să obțină punctajul maxim, întrucât va depăși considerabil limita de timp alocată.

- Pentru a obține punctajul maxim, vom alege să calculăm într-un mod mai eficient matricele $Nord[][]$, $Sud[][]$, $Vest[][]$, $Est[][]$. Ne propunem ca fiecare linie din aceste matrice să fie corect calculată folosind doar $O(N)$ pași, reducând astfel, complexitatea totală la $O(N \cdot N)$.

- Pentru a calcula cum va arăta linia i din matricea $Est[][]$, vom folosi un vector auxiliar, procedând astfel:

```
int K = 0;

for(int j = 1; j <= N; ++j)
{
    while(K && H[i][j] > H[i][V[K]])
        Est[i][V[K]] = j, --K;

    V[++K] = j;
}

for(int j = 1; j <= K; ++j)
    Est[i][V[j]] = (N + 1);
```

- Pentru a nu degenera complexitatea totală $O(N * N)$, marcarea pomilor supravegheați de dispozitive se va face utilizând https://www.infoarena.ro/multe-smenuri-de-programare-in-cc-si-nu-numai *Șmenul lui Mars*.

2.3 Problema Puternic

Propunător: prof. Ana-Maria Arișanu
Colegiul Național "Mircea cel Bătrân", Rm. Vâlcea

Enunț

Un număr *puternic* este un număr natural mai mare decât 1 care are proprietatea că dacă este divizibil cu numărul prim p atunci este divizibil și cu p^2. De exemplu, 36 și 27 sunt numere puternice, în timp ce 12 nu este număr puternic deoarece este divizibil cu 3 și nu este divizibil cu 3^2. La ora de matematică, elevii au aflat ce înseamnă un număr puternic. Pentru a verifica dacă elevii au înțeles, domnul profesor a scris pe tablă un șir de N numere naturale, $X_1, X_2, X_3, \ldots, X_N$ și l-a rugat pe Mihai să șteargă din șir numerele puternice.

Analizând numerele rămase, Mihai a observat că se poate obține un număr puternic prin *concatenarea* a două numere din șirul rămas, numere egal depărtate de capetele acestui nou șir. *Concatenarea* presupune lipirea numărului din a doua jumătate a șirului la finalul celui aflat în prima jumătate. Dacă noul șir are număr impar de elemente, elementul din mijloc se ignoră. De exemplu, dacă șirul obținut după ștergerea numerelor puternice este: 12, 1, 19, 13, 3, 21, 5 atunci numerele obținute prin concatenare sunt 125, 121, 193.

Cerințe

Scrieți un program care citește un număr natural N și apoi un șir de N numere naturale și determină:

1. Câte numere puternice sunt în șirul dat;
2. Care sunt perechile de numere din șirul rămas după ștergerea numerelor puternice, numere egal departate de capetele șirului, prin concatenarea cărora se obține un număr puternic.

Date de intrare

Fişierul de intrare **puternic.in** conţine pe prima linie un număr natural C. Pentru toate testele de intrare, numărul C poate avea doar valoarea 1 sau 2. Pe a doua linie a fișierului se găsește numărul natural N. Pe a treia linie se găsesc N numere naturale separate prin câte un spațiu.

Date de ieșire

Dacă $C = 1$, se va rezolva cerința 1. În acest caz, fişierul de ieşire **puternic.out** va conține pe prima linie un număr natural reprezentând numărul de numere puternice din șirul dat.
Dacă $C = 2$, se va rezolva cerința 2. În acest caz, fişierul de ieşire **puternic.out** va conține perechile de numere egal depărtate de capetele șirului obținut după ștergere, prin concatenarea cărora de obține un număr puternic. Fiecare pereche se scrie pe câte un rând, iar numerele din pereche se scriu separate printr-un spațiu, primul număr fiind cel din stânga. Dacă sunt mai multe astfel de perechi se vor afișa, în ordine, începând cu cea apropiată de capetele noului șir. Dacă nu există nicio astfel de pereche, în fișierul puternic.out se va afișa -1.

Restricții și precizări

- $1 \leq N \leq 100\ 000$
- $1 \leq X_1, X_2, X_3, \ldots, X_N \leq 1\ 000\ 000\ 000$
- Pentru rezolvarea corectă a primei cerințe se obțin 30 de puncte, iar pentru rezolvarea corectă a celei de a doua cerințe se obțin 70 de puncte.

Exemple

1)	`puternic.in`	`puternic.out`	**Explicații**
	1 8 100 28 16 11 484 25 162 27	5	Se rezolvă cerința **1**. Numerele puternice sunt 100 16 484 25 27
1)	`puternic.in`	`puternic.out`	**Explicații**
	2 11 12 9 1 8 19 6 3 4 49 21 5	12 5 1 21	Se rezolvă cerința **2**. Se va folosi doar valoarea N. După ștergerea numerelor puternice, șirul rămas este: 12 1 19 6 3 21 5.

Descrierea soluției

Cerința 1

Descompunem în factori primi fiecare număr **mai mare decât 1** din șir pentru a verifica dacă este puternic.

Dacă prin descompunere obținem un factor prim la puterea 1, numărul respectiv nu este puternic.

Cerința 2

Memorăm într-un vector numerele din șir care nu sunt puternice.

Concatenăm numerele egal depărtate de capetele noului șir și verificăm dacă numărul obținut prin concatenare este puternic. Numerele din șirul inițial sunt $\leq 10^9$, prin urmare în urma concatenării se obțin numere $\leq 10^{18}$ (long long).

Pentru a verifica eficient dacă un număr x este puternic ținem cont de următoarele observații matematice:

- $4000^5 = 1.024.000.000.000.000.000 \approx 10^{18}$
- numărul x trebuie împărțit, de câte ori este posibil, la toate numere prime mai mici sau egale cu 4000. Dacă în timpul acestei descompuneri obținem un factor prim la putere 1, atunci numărul x nu este puternic
- După împărțiri rămânem cu x fie 1 (și e puternic), fie produs de numere prime **mai mari decât 4000**. Mai exact x poate fi produs de până la 4 numere prime mai mari ca 4000:
  - dacă este număr prim NU poate fi puternic
  - dacă este produs de 4 numere prime, pentru a fi puternic, poate fi de forma p^4 sau $p^2 * q^2$ (adică pătrat perfect, indiferent de formă)

- daca e produs de 3 numere prime pentru a fi puternic trebuie sa fie de forma p^3 (trebuie sa se divida cu p si cu p^2).
- daca e produs de 2 numere prime pentru a fi puternic trebuie sa aibă forma p^2

Putem folosi Ciurul lui Eratostene până la 4000, pentru a verifica dacă un număr obținut prin concatenare este puternic.

Pentru a verifica dacă numărul rămas este pătrat perfect sau cub perfect putem folosi o căutare binara ori biblioteca *cmath* și calcula direct rădăcina pătrată (sqrt) și radical de ordin 3(cbrt) (atenție la erorile de precizie, se poate că uneori $sqrt(16) = 3.99999$).

Capitolul 3

Clasa a VII-a

3.1 Problema Cat2Pal

Propunător: prof. Ionel-Vasile Piţ-Rada

Colegiul Naţional "Traian", Drobeta-Turnu Severin

Enunț

Prin concatenarea a două numere naturale A și B se pot obține numerele naturale AB și BA. De exemplu, dacă $A = 8$ și $B = 8$, atunci prin concatenare se poate obține numărul 88, iar dacă $A = 7$ și $B = 17$, atunci prin concatenare se pot obține numerele 717 și respectiv 177.

Cerințe

Scrieţi un program care să rezolve următoarele două cerinţe:

1. Pentru un număr natural nenul A dat, să se calculeze P_1, numărul numerelor naturale distincte X, unde $1 \leq X \leq 10 \cdot A$, astfel încât X concatenat cu A sau A concatenat cu X este palindrom.

2. Date fiind numărul natural N și un șir de N numere naturale $v[1], v[2], \ldots, v[N]$, să se calculeze P_2, numărul de numere palindrom distincte care se pot obține prin concatenarea numerelor din perechile $(v[i], v[j])$, unde $1 \leq i \leq N$ și $1 \leq j \leq N$.

Date de intrare

Fișierul de intrare *cat2pal.in* conţine pe prima linie numărul natural C, reprezentând cerinţa care urmează să fie rezolvată (1 sau 2).

Dacă $C = 1$, atunci pe linia a doua se află numărul natural A.

Dacă $C = 2$, atunci pe linia a doua se află numărul natural N și pe linia a treia se află N numere naturale separate prin spațiu, reprezentând șirul v.

Date de ieșire

Fișierul de ieşire *cat2pal.out* va conţine o singură linie pe care se va scrie un singur număr natural, reprezentând rezultatul pentru cerinţa C din fişierul de intrare.

Restricții și precizări

- $1 \leq A < 100000000$
- $1 \leq N \leq 10000$
- $0 \leq v[i] < 100000, \forall\ i \in \{1, 2, \ldots, N\}$
- Pentru cazul $C = 2$, trebuie luate în considerare și perechile $(v[i], v[i])$, adică concatenarea unui element cu el însuși.
- Pentru teste valorând 20 de puncte: $C = 1$ şi $A < 100000$
- Pentru alte teste valorând 30 de puncte: $C = 1$ şi nu există restricții suplimentare.
- Pentru teste valorând 15 puncte: $C = 2$ și $N \leq 1000$
- Pentru teste valorând 35 puncte: $C = 2$ şi nu există restricții suplimentare.

Exemple

1)	`cat2pal.in`	`cat2pal.out`	**Explicații**
	1 2	3	Se rezolvă cerința **1**. **N = 9**. $A = 2$, numerele X care concatenate cu A produc numere palindrom sunt: $2, 12$ şi 20, deci $P_1 = 3$.
2)	`cat2pal.in`	`cat2pal.out`	**Explicații**
	2 3 2 12 21	4	Se rezolvă cerința **2**. $N = 3$, $v = \{2, 12, 21\}$, numerele palindrom distincte care pot fi obținute sunt: $22, 212, 1221, 2112$, deci $P_2 = 4$.

Descrierea soluției

Soluție brute-force – 35 puncte

Prima cerință se poate rezolva printr-o metoda brute-force. Putem itera X prin toate numerele de la 1 la $10 \cdot A$ și verifica prin construcția numerelor $X|A$ și $A|X$ (prin $|$ am notat operația de concatenare). Se pot obține astfel 20 puncte, în funcție și de constanta implementării.

Complexitate temporală: $\mathcal{O}(A \cdot log(A))$

A doua cerință se poate rezolva tot folosind o metodă brute-force. Se pot analiza toate perechile de valori date. Se pot obține astfel 15 puncte, în funcție și de constanta implementării.

Complexitate temporală: $\mathcal{O}(N^2 \cdot log(MaxVal))$

Soluție oficială – 100 de puncte

Pentru prima cerință se poate observa că este suficient a fi analizate:

- pentru concatenările $X|A$ prefixele oglinditului numărului A și prefixele concatenate la dreapta cu o cifra $\{0, 1, 2, \ldots, 9\}$
- pentru concatenările $A|X$ sufixele oglinditului numărului A și sufixele concatenate la stânga cu o cifra $\{1, 2, \ldots, 9\}$

Astfel se pot depune aceste numere într-un vector și apoi se verifică și se numără valorile distincte care corespund cerințelor. **Complexitate temporală:** $\mathcal{O}(log^2(A))$

Pentru a doua cerință se vor parcurge pentru fiecare valoare (asemănător cu rezolvarea de la prima cerință) prefixele și sufixele valorii oglindite și pentru fiecare dintre acestea se va încerca obținerea unui palindrom. Pentru fiecare palindrom obținut se va construi un sufix identificator, pentru palindrom cu k cifre sufixul va avea $k - \frac{k}{2}$ cifre, care va fi marcat într-un vector de poziție. La final vor fi numărate pozițiile marcate.

Complexitate temporală: $\mathcal{O}(N \cdot log^2(MaxVal))$

3.2 Problema Virus

Propunător: prof. Adrian Pintea
Colegiul Național "Andrei Mureşanu", Dej

Enunț

Un laborator specializat studiază mutațiile unui virus pandemic pentru a găsi cel mai bun vaccin pentru combaterea acestuia.

Codul unui virus este un şir format din litere (mari şi mici) ale alfabetului englez. Numim *mutație* a virusului pandemic un şir de caractere care are aceeași lungime cu codul virusului şi care conţine o singură poziţie pentru care litera din şir este diferită de litera situată pe poziţia respectivă în codul virusului pandemic.

De exemplu, pentru virusul pandemic având codul **abac**, șirul **Bbac** reprezintă o mutaţie, deoarece are aceeaşi lungime şi diferă doar prin litera de pe prima poziție.

Laboratorul primeşte o listă conţinând codurilor mai multor viruși descoperiți în urma testărilor.

Cerințe

Scrieţi un program care, cunoscând codul virusului pandemic şi lista codurilor viruşilor descoperiţi în urma testărilor, rezolvă următoarele cerinţe:

1. Determină numărul de mutații ale virusului pandemic existente în listă, mutații nu neapărat distincte;

2. Determină mutația cu număr maxim de apariții în listă; dacă există mai multe mutaţii cu acelaşi număr maxim de apariţii, se va determina prima mutaţie, în ordine lexicografică.

Date de intrare

Fişierul de intrare *virus.in* conţine pe prima linie un număr natural C reprezentând cerinţa care trebuie să fie rezolvată (1 sau 2). Pe cea de a doua linie se află codul virusului pandemic. Pe a treia linie se află un număr natural N, reprezentând numărul de viruşi existenţi în lista primită de laborator. Pe următoarele N linii se află codurile viruşilor din listă, câte un cod pe o linie.

Date de ieșire

Fişierul de ieşire *virus.out* va conţine o singură linie:

- Dacă $C = 1$, pe prima linie va fi scris un număr natural care reprezintă câte elemente din listă sunt mutații ale virusului pandemic.

- Dacă $C = 2$, pe prima linie va fi scris un şir de caractere care reprezintă mutația cu număr maxim de apariții. Dacă există mai multe mutaţii cu număr maxim de apariţii, va fi afişată prima (cea mai mică), în ordine lexicografică.

Restricții și precizări

- $2 \leq N \leq 50000$
- Lungimea maximă a codului unui virus este 200
- Daca a și b sunt două siruri de lungime lg, spunem că șirul $a = a_0a_1a_2a_{lg-1}$ este mai mic din punct de vedere lexicografic decât şirul $b = b_0b_1b_2b_{lg-1}$, dacă există o poziţie $k \in \{0, 1, \ldots, lg-1\}$ astfel încât $a_i = b_i$ pentru orice $0 \leq i < k$ și $a_k < b_k$.
- În codul ASCII codurile literelor mari sunt mai mici decât codurile literelor mici.
- Pentru teste valorând 31 de puncte: $C = 1$
- Pentru alte teste valorând 16 de puncte: $C = 2$, $N \leq 500$ și lungimea maxima a codului unui virus este 40.
- Pentru alte teste valorând 53 de puncte: $C = 2$ și nu există restricții suplimentare.

Exemple

1)	virus.in	virus.out	**Explicații**
	1 abac 5 Abbbq Zbac abbC aBac Zbac	3	Se rezolvă cerința **1**. Mutaţiile sunt **Zbac**, **aBac** și **Zbac**.
2)	**virus.in**	**virus.out**	**Explicații**
	2 abcD 8 abcdD XbcD Xc XbcD aXcD aXcD aXc ab	XbcD	Se rezolvă cerința **2**. Mutațiile **XbcD** și **aXcD** apar fiecare de câte 2 ori. Prima mutație dintre acestea în ordine lexicografică este **XbcD**. **XbcD** este mai mic lexicografic decât **aXcD**, deoarece în codul ASCII: 'A' <'B' <...<'Z' <'a' <'b' <...<'z' .

Descrierea soluției

Cerința 1 (20 de puncte)

O soluție, cu complexitate $\mathcal{O}(\mathbf{N})$ se poate obține astfel: odată cu citirea elementelor șirului $\mathbf{v[1]}, \mathbf{v[2]}, ..., \mathbf{v[N]}$ se construiesc sumele parțiale $\mathbf{S[k]} = \mathbf{v[1]} + \mathbf{v[2]} + ... + \mathbf{v[k]} = \mathbf{S[i-1]} + \mathbf{v[k]}$. Se va utiliza un vector de frecvență **Rest**[] a valorilor resturilor împărțirilor acestor sume parțiale la **N**. La fiecare rest negativ se va aduna **N**, rezultând un rest pozitiv din mulțimea $\{0, 1, 2, \ldots, \mathbf{N}-1\}$.

Cerința 1 (31 puncte)

Se vor determina numărul de mutații ale virusului pandemic existente în listă, mutații nu neapărat distincte după următorul algoritm:

- Citim codul virusul pe care îl notăm cu v și cu $nr-$ numărul de mutații (inițial este 0).
- Citim codurile pe rând din fișier și comparăm fiecare cod citit cu v, numărând pozițiile pe care se afla caractere distincte. În cazul în care codul citit are aceeași lungime egală cu lungimea codului virusului v și diferă printr-o singură poziție, incrementăm nr.
- La final scriem numărul nr în fișierul de ieșire.

Complexitate temporală: $\mathcal{O}(N \cdot L_{max})$

Cerința 2 (69 puncte)

Soluția brute-force – 16 puncte

Putem itera prin toate codurile date, iar dacă un cod este mutație, iterăm din nou prin toate codurile pentru a număra de câte ori apare această mutație în listă. Dacă găsim o mutație care apare de mai multe ori sau apare de la fel de multe ori ca cel mai bun răspuns de până acum, dar este mai mică din punct de vedere lexicografic, o actualizăm.

Complexitate temporală: $\mathcal{O}(N^2 \cdot L_{max})$

Soluția oficială

Pentru a obține o complexitate mai bună vom construi un tabel bidimensional fr cu maxim L linii (numărul de litere din codul virusului) și 52 coloane (asociate literelor mici și mari).

La citirea unui cod din fișier se verifică dacă este mutație și în caz afirmativ incrementăm $fr[pos][ch]$, unde pos este poziția pe care apare litera diferită față de virusul v, iar ch este litera prin care diferă față de virusul v

Pentru determinarea maximului se parcurge tabelul bidimensional și se ține cont la modificarea valorii maxime și de ordinea lexicografică pentru mutație.

Complexitate temporală: $\mathcal{O}(N \cdot L_{max} + {L_{max}}^2 \cdot \Sigma)$

3.3 Problema Zid

Propunător: prof. Nistor Moţ
Şcoala "Dr. Luca", Brăila

Enunț

Un zid ornamental de formă dreptunghiulară este alcătuit din N rânduri de cărămizi, fiecare rând având câte M cărămizi identice, aşezate una lângă alta. Fiecare cărămidă este colorată într-una dintre culorile $\{0, 1, 2, \ldots, Cmax\}$.
Un *pătrat* de latură L în acest zid este constituit din cărămizile situate pe L rânduri consecutive şi L coloane consecutive.
Spunem că un pătrat este *colorat uniform* dacă el conţine acelaşi număr de cărămizi din fiecare culoare care apare în pătratul respectiv.

Cerință

Scrieţi un program care, cunoscând configuraţia zidului, determină în acest zid un pătrat de latură maximă, colorat uniform.

Date de intrare

Fişierul de intrare *zid.in* conţine pe prima linie numerele naturale N M $Cmax$, reprezentând numărul de rânduri de cărămizi, numărul de cărămizi de pe fiecare rând, respectiv culoarea maximă.
Pe următoarele N linii este descrisă configuraţia zidului, de sus în jos; pe fiecare linie dintre cele N se află câte M numere naturale, reprezentând culorile cărămizilor de pe rândul respectiv, în ordine, de la stânga la dreapta. Valorile scrise pe aceeaşi linie sunt separate prin câte un spaţiu.

Date de ieșire

Fişierul de ieşire *zid.out* va conţine o singură linie pe care vor fi scrise 3 numere naturale Nr R C, separate prin câte un singur spaţiu, reprezentând numărul de cărămizi existente într-un pătrat colorat uniform de latură maximă, respectiv rândul şi cărămida de pe rând situată în colţul din stânga-sus al pătratului colorat uniform de latură maximă.

Restricții și precizări

- $2 \leq N, M \leq 250$
- $1 \leq Cmax \leq 9$
- Rândurile sunt numerotate de sus în jos de la 1 la N. Cărămizile situate pe un rând sunt numerotate de la stânga la dreapta de la 1 la M.
- Dacă există mai multe pătrate colorate uniform de latură maximă se va alege pătratul pentru care numărul rândului este minim. Dacă există mai multe pătrate colorate uniform de latură maximă care au colţul din stânga-sus pe acelaşi rând minim, se va alege pătratul cel mai din stânga.
- Pentru teste valorând 22 de puncte: $2 \leq N, M \leq 30$
- Pentru alte teste valorând 23 de puncte: $30 < N, M \leq 100$

- Pentru alte teste valorând 10 de puncte, cărămizile sunt vopsite doar în două culori: $Cmax = 1$
- Pentru alte teste valorând 45 de puncte: $100 < N, M \leq 250$

Exemplu

	zid.in	zid.out	Explicații
	6 8 5 1 2 3 5 1 2 3 5 1 2 1 2 3 5 3 5 1 1 1 2 2 3 3 3 1 2 3 5 5 3 2 1 3 3 1 1 2 2 5 5 2 1 2 3 2 5 2 2	9 2 4	Pătratul colorat uniform de latură maximă, situat pe rândul cel mai de sus, cel mai în stânga este: 2 3 5 2 2 3 5 5 3 El conţine 9 cărămizi, în care apar culorile $2, 3, 5$ de câte 3 ori fiecare.Acest pătrat are colţul din stânga-sus situat pe rândul al doilea, în a patra cărămidă de pe rând(a patra coloană). Există şi alte pătrate colorate uniform formate din 9 cărămizi, de exemplu: 1 2 3 3 3 1 2 1 2 dar acesta are colţul din stânga-sus pe rândul 4.

Descrierea soluției

Vom memora configuraţia zidului într-o matrice Z cu N linii şi M coloane, unde $Z[i][j]$ reprezintă culoarea cărămizii din poziţia (i, j) în zid.

Soluția 1 – 22 de puncte

Parcurgem toate pătratele posibile şi verificăm pentru fiecare pătrat dacă este uniform colorat, parcurgând pătratul şi contorizând culorile care apar în pătrat într-un vector de frecvenţă.

Pentru a construi toate pătratele posibile fixăm colţul stânga-sus al pătratului în toate modurile posibile, apoi fixăm latura pătratului în toate modurile posibile (preferabil în ordine descrescătoare, astfel încât prima soluţie găsită să fie de latură maximă).

Iterarea prin toate pătratele are complexitatea $\mathcal{O}(N \cdot M \cdot min(N, M))$, iar pentru fiecare pătrat trebuie să facem şi o verificare prin parcurgerea pătratului.

În final, **complexitatea temporală** este: $\mathcal{O}(N \cdot M \cdot min(N, M) \cdot (N \cdot M + C))$.

Soluția 2 – 45 de puncte

Optimizăm soluția 1, încercând să evităm să parcurgem întreg pătratul pentru a calcula frecvenţele culorilor.

Construim un tablou tridimensional Fr, unde $Fr[i][j][k]$ reprezintă numărul de apariţii ale culorii k pe rândul i al zidului, considerând primele j cărămizi.

Precalculând acest tablou, putem afla frecvenţa de apariţie a fiecărei culori într-un pătrat de latură L şi colţul stânga-sus în poziţia (i, j) parcurgând doar liniile de la i la $i + L - 1$ şi determinând frecvenţele de apariţie ale culorilor pentru coloanele $j, \ldots, j + L - 1$ prin scădere

(scădem $Fr[x][j+L-1][k] - Fr[x][j-1][k]$ pentru a determina frecvenţa de apariţie a culorii k pe linia x).

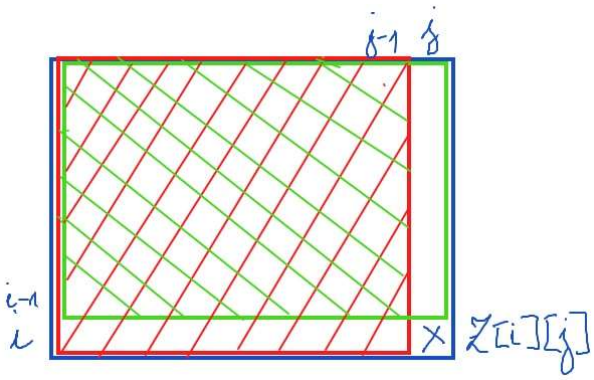
Complexitatea temporală: $\mathcal{O}(N \cdot M \cdot min(N, M) \cdot M \cdot C)$.

Soluția oficială – 100 de puncte

Optimizăm în continuare soluția 2, pentru a determina frecvenţa de apariţie a unei culori într-un pătrat în complexitate $\mathcal{O}(1)$.

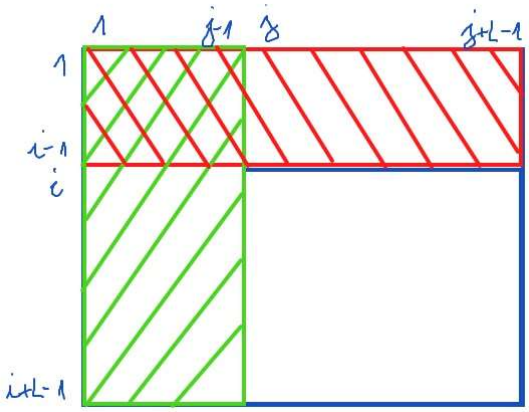
Pentru aceasta vom construi un tablou tridimensional Fr, unde $Fr[i][j][k]$ reprezintă numărul de apariţii ale culorii k în submatricea cu colţul stânga-sus în poziţia $(1, 1)$ şi colţul dreapta-jos în poziţia (i, j). Precalcularea acestui tablou se poate face în $\mathcal{O}(N \cdot M \cdot C)$ astfel:

Pentru $i \leftarrow 1,\ N$ execută
  Pentru $j \leftarrow 1,\ M$ execută
    Pentru $k \leftarrow 0,\ C$ execută
      $Fr[i][j][k] = Fr[i-1][j][k] + Fr[i][j-1][k] - Fr[i-1][j-1][k] + (Z[i][j] == k)$



Pentru a memora frecvenţa de apariţie a fiecărei litere într-un pătrat cu colţul în stânga-sus în poziţia (i, j) şi colţul din dreapta jos în poziţia $(i + L - 1, j + L - 1)$ vom utiliza un vector uz, unde $uz[k]$ reprezintă frecvenţa de apariţie a culorii k. Calcularea vectorului uz se poate face în $\mathcal{O}(C)$, în următorul mod:

Pentru $k \leftarrow 0,\ C$ execută
  $uz[k] = F[i+L-1][j+L-1][k] - F[i+L-1][j-1][k] - F[i-1][j+L-1][k] + F[i-1][j-1][k]$



Complexitatea temporală: $\mathcal{O}(N \cdot M \cdot min(N, M) \cdot C)$.

Capitolul 4

Clasa a VIII-a

4.1 Problema Bile

Propunător: stud.drd. Diana Ghinea

Eidgenössische Technische Hochschule (ETH) Zürich

Enunț

Presupunem că avem două cutii notate **A** și **B**. Cutia **A** conține **N** bile numerotate cu numerele naturale distincte: $\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots, \mathbf{N-1}$. Cutia **B** este goală.

Spunem că o bilă dintr-o cutie este bila **specială** a acestei cutii dacă numărul **X** cu care este numerotată această bilă este egal cu media aritmetică a numerelor celorlalte bile din cutie.

La un moment dat, cineva mută bila cu numărul **K** din cutia **A** în cutia **B**.

Vi se cere să alegeți alte **K** bile, din cutia **A**, pe care să le mutați în cutia **B** astfel încât cutia **B** să conțină $\mathbf{K+1}$ bile, iar bila cu numărul **K** să fie bila specială a cutiei **B**.

Cerințe

Scrieți un program care citește numerele **N** și **K**, apoi determină:

1. dacă, înainte să fie mutate bile din cutia **A** în cutia **B**, există o bilă specială în cutia **A**; în caz afirmativ, programul determină numărul **X** cu care este numerotată această bilă specială;
2. **cel mai mic** (în sens lexicografic) **șir strict crescător** al numerelor celor **K** bile care pot fi mutate din cutia **A** în cutia **B** astfel încât bila cu numărul **K** să fie bila specială a cutiei **B**;
3. **cel mai mare** (în sens lexicografic) **șir strict crescător** al numerelor celor **K** bile care pot fi mutate din cutia **A** în cutia **B** astfel încât bila cu numărul **K** să fie bila specială a cutiei **B**.

Date de intrare

Fișierul de intrare `bile.in` conține pe prima linie trei numere naturale **C**, **N** și **K**, separate prin câte un spațiu. **C** reprezintă cerința care trebuie rezolvată (**1**, **2** sau **3**), iar **N** și **K** au semnificația din enunț.

Date de ieșire

Fişierul de ieşire `bile.out` va conţine:

- dacă $\mathbf{C = 1}$, pe prima linie, numărul natural $\mathbf{X}$ reprezentând numărul bilei speciale din cutia A sau valoarea $\mathbf{-1}$ dacă cutia A nu conține o astfel de bilă (reprezentând răspunsul la cerinţa **1**);
- dacă $\mathbf{C = 2}$, pe prima linie, **un șir strict crescător de K** numere naturale, separate prin câte un spațiu (reprezentând răspunsul la cerința **2**);
- dacă $\mathbf{C = 3}$, pe prima linie, **un șir strict crescător de K** numere naturale, separate prin câte un spațiu (reprezentând răspunsul la cerința **3**).

Restricții și precizări

- $\mathbf{N}$ număr natural, $\mathbf{4 \leq N \leq 100\ 000}$
- $\mathbf{K}$ număr natural $\mathbf{2 \leq K \leq N/2}$
- Șirul $\mathbf{y_1, y_2, \ldots, y_K}$ este **mai mic în sens lexicografic** decât șirul $\mathbf{z_1, z_2, \ldots, z_K}$ dacă există un indice $\mathbf{p}$, $\mathbf{1 \leq p \leq K}$, **astfel încât:** $\mathbf{y_1 = z_1, y_2 = z_2, \ldots, y_{p-1} = z_{p-1}}$ și $\mathbf{y_p < z_p}$
- Pentru cerința 1 se acordă $20p$, iar pentru fiecare dintre cerințele 2 și 3 se acordă câte $40p$.

Exemple

1)	`bile.in`	`bile.out`	**Explicații**
	1 9 3	4	Se rezolvă cerința **1**. $\mathbf{N = 9}$. Avem **9** bile inscripționate cu $\mathbf{0, 1, 2, 3, 4, 5, 6, 7, 8}$. Bila specială este $\mathbf{X = 4}$ deoarece: $\mathbf{X = (0 + 1 + 2 + 3 + 5 + 6 + 7 + 8)/8 = 32/8 = 4}$.
2)	`bile.in`	`bile.out`	**Explicații**
	1 8 3	-1	Se rezolvă cerința **1**. $\mathbf{N = 8}$. Se va scrie în fișierul de ieșire valoarea $\mathbf{-1}$ deoarece cutia **A** nu conține nicio bilă specială.
3)	`bile.in`	`bile.out`	**Explicații**
	2 8 3	0 2 7	Se rezolvă cerința **2**. $\mathbf{N = 8}$. Șirurile strict crescătoare ale numerelor bilelor care pot fi mutate în cutia **B**, lângă bila specială $\mathbf{K = 3}$, sunt: $\mathbf{0, 2, 7}$ sau $\mathbf{0, 4, 5}$ sau $\mathbf{1, 2, 6}$, deoarece: $\mathbf{3 = (0 + 2 + 7)/3 = (0 + 4 + 5)/3 = (1 + 2 + 6)/3}$. Cel mai mic șir **în sens lexicografic**, crescător, este: $\mathbf{0, 2, 7}$.
4)	`bile.in`	`bile.out`	**Explicații**
	3 8 3	1 2 6	Se rezolvă cerința **3**. $\mathbf{N = 8}$. Șirurile strict crescătoare ale numerelor bilelor care pot fi mutate în cutia **B**, lângă bila specială $\mathbf{K = 3}$, sunt: $\mathbf{0, 2, 7}$ sau $\mathbf{0, 4, 5}$ sau $\mathbf{1, 2, 6}$, deoarece: $\mathbf{3 = (0 + 2 + 7)/3 = (0 + 4 + 5)/3 = (1 + 2 + 6)/3}$. Cel mai mare șir **în sens lexicografic**, crescător, este: $\mathbf{1, 2, 6}$.

Descrierea soluției

Cerința 1 (20 puncte)

Notăm cu $\mathbf{S}$ suma numerelor bilelor din cutia $\mathbf{A}$: $\mathbf{S} = 0 + 1 + \cdots + (\mathbf{N}-1) = \frac{\mathbf{N}(\mathbf{N}-1)}{2}$. Fie $\mathbf{X}$ numărul bilei speciale din cutia $\mathbf{A}$. Atunci, $\mathbf{X} = (\mathbf{S} - \mathbf{X})/(\mathbf{N}-\mathbf{1})$, de unde rezultă că $\mathbf{X} = (\mathbf{N}-\mathbf{1})/\mathbf{2}$. Astfel, cutia $\mathbf{A}$ conține o bilă specială doar dacă $\mathbf{2} \mid (\mathbf{N}-\mathbf{1})$, deci dacă $\mathbf{N}$ este impar.

Complexitatea soluției: $\mathcal{O}(1)$.

Cerința 2 (40 puncte)

- Dacă $\mathbf{K^2} - \frac{\mathbf{(K-2)(K-1)}}{\mathbf{2}} \leq \mathbf{N}-\mathbf{1}$, atunci răspunsul pentru cerința 2 este șirul de numere:

$$\underbrace{0, 1, 2, 3, \ldots, \mathbf{K}-2,}_{\substack{\text{cele mai mici } \mathbf{K}-1 \\ \text{numere naturale distincte}}} \mathbf{K}^2 - \frac{(\mathbf{K}-2)(\mathbf{K}-1)}{2}.$$

- Dacă $\mathbf{K^2} - \frac{\mathbf{(K-2)(K-1)}}{\mathbf{2}} > \mathbf{N}-\mathbf{1}$, notăm soluția acestei cerințe cu s.

  Inițializăm $s[i] = i$ pentru $i = 0, 1, \ldots, \mathbf{K}-1$ și notăm cu S suma valorilor inițiale din s: $S = s[0] + s[1] + \cdots + s[\mathbf{K}-1] = \mathbf{K}(\mathbf{K}-1)/2$.

  Pentru ca media aritmetică a șirului s să fie $\mathbf{K}$, trebuie să distribuim diferența $dif = \mathbf{K}^2 - S$ la elementele din șir. Întrucât dorim să obținem cea mai mică soluție din punct de vedere lexicografic, vom utiliza această diferență pentru a maximiza ultimele elemente ale lui s.

  Șirul s trebuie să conțină numere distincte mai mici sau egale cu $\mathbf{N}-1$. Astfel, cea mai mare valoare posibilă pentru $s[\mathbf{K}-1]$ este $\mathbf{N}-1$, cea mai mare valoare posibilă pentru $s[\mathbf{K}-2]$ este $\mathbf{N}-2$, ..., iar cea mai mare valoare posibilă pentru $s[0]$ este $\mathbf{N}-\mathbf{K}$. Deci, diferența dintre valoarea actuală a oricărui element din s și valoarea maximă a acelui element este $\mathbf{N}-\mathbf{K}$.

  Întrucât putem distribui doar diferența dif, adăugăm la ultimele $g = [dif/(\mathbf{N}-\mathbf{K})]$ elemente din șir cantitatea $\mathbf{N}-\mathbf{K}$. Rămâne să adăugăm restul $rest = dif \ \% \ (\mathbf{N}-\mathbf{K})$. Încercăm să adăugăm acest rest la elementul $s[\mathbf{K}-(g+1)]$. Dacă $s[\mathbf{K}-(g+1)]+rest \neq \mathbf{K}$, adăugăm restul astfel:

Indice i	0	1	2	...	$\mathbf{K}-g-2$	$\mathbf{K}-g-1$	$\mathbf{K}-g$	$\mathbf{K}-g+1$	...	$\mathbf{K}-2$	$\mathbf{K}-1$
s[i]	0	1	2	...	$\mathbf{K}-g-2$	$\mathbf{K}-g-1+rest$	$\mathbf{N}-g$	$\mathbf{N}-g+1$		$\mathbf{N}-2$	$\mathbf{N}-1$

  Altfel, dacă $s[\mathbf{K}-(g+1)] + rest = \mathbf{K}$, atunci șirul se modifică astfel:

Indice i	0	1	2	...	$\mathbf{K}-g-2$	$\mathbf{K}-g-1$	$\mathbf{K}-g$	$\mathbf{K}-g+1$	...	$\mathbf{K}-2$	$\mathbf{K}-1$
s[i]	0	1	2	...	$\mathbf{K}-g-2$	$\mathbf{K}-g+rest$	$\mathbf{N}-g-1$	$\mathbf{N}-g+1$		$\mathbf{N}-2$	$\mathbf{N}-1$

Complexitatea soluției: $\mathcal{O}(\mathbf{K})$.

Cerința 3 (40 puncte)

Notăm soluția acestei cerințe cu s. Întrucât s trebuie să fie cea mai mare soluție din punct de vedere lexicografic, încercăm să maximizăm primele elemente din s. Putem construi s astfel:

- **K par**
  Fie $\mathbf{M} = \mathbf{K}/2$. Cea mai mare valoare posibilă pentru $s[0]$ astfel încât media valorilor din șir să fie $\mathbf{K}$ este $\mathbf{K}/2$. Alegem valorile celorlalți termeni din șir astfel încât:
  - suma termenilor situați pe poziții simetrice față de mijlocul șirului să fie egală cu $2\mathbf{K}$:
    $s[0] + s[\mathbf{K}-1] = 2\mathbf{K}, s[1] + s[\mathbf{K}-2] = 2\mathbf{K}, \ldots, s[\mathbf{M}-1] + s[\mathbf{M}] = 2\mathbf{K}$.

  Deci, pentru $\mathbf{K}$ par, obținem următoarea soluție.

Indice i	0	1	2	. . . . .	$\mathbf{M}-1$	$\mathbf{M}$	$\mathbf{M}+1$	. . . . .	$\mathbf{K}-1$
s[i]	$\mathbf{K}/2$	$\mathbf{K}/2+1$	$\mathbf{K}/2+2$	. . . . .	$\mathbf{K}-1$	$\mathbf{K}$	$\mathbf{K}+2$	. . . . .	$3 \cdot \mathbf{K}/2$

- **K impar**
  Fie $\mathbf{M} = [\mathbf{K}/2]$. Cea mai mare valoare posibilă pentru $s[0]$ astfel încât media valorilor din șir să fie $\mathbf{K}$ este $[\mathbf{K}/2]$. Alegem valorile celorlalți termeni din șir astfel încât:
  - $s[0] + s[\mathbf{K}-1] = 2\mathbf{K}+1$;
  - $s[\mathbf{M}] = \mathbf{K}-1$;
  - suma termenilor situați pe poziții simetrice față de mijlocul șirului să fie egală cu $2\mathbf{K}$:
    $s[1] + s[\mathbf{K}-2] = 2\mathbf{K}, s[2] + s[\mathbf{K}-3] = 2\mathbf{K}, \ldots, s[\mathbf{M}-1] + s[\mathbf{M}+1] = 2\mathbf{K}$;

  Deci, pentru $\mathbf{K}$ impar, obținem următoarea soluție.

Indice i	0	1	2	...	$\mathbf{M}-1$	$\mathbf{M}$	$\mathbf{M}+1$	...	$\mathbf{K}-1$
s[i]	$[\mathbf{K}/2]$	$[\mathbf{K}/2] + \mathbf{1}$	$[\mathbf{K}/2]+2$	...	$\mathbf{K}-2$	$\mathbf{K}-1$	$\mathbf{K}+2$	...	$2\mathbf{K}+1-[\mathbf{K}/2]$

Complexitatea soluției: $\mathcal{O}(\mathbf{K})$.

Observații

- Nu este necesar să se utilizeze un tablou unidimensional pentru a memora termenii lui s.
- O soluție care generează toate modalitățile de a scrie pe $\mathbf{K}^2$ ca o suma de $\mathbf{K}$ numere cu proprietatea cerută va obține un punctaj parțial.

4.2 Problema Secvențe

Propunător: ș.l. Stelian Ciurea
Universitatea "Lucian Blaga" din Sibiu

Enunț

Se consideră un șir cu **N** elemente numere întregi. Definim următoarele noțiuni:

- secvență în șir = elemente situate pe poziții consecutive în șir
- lungimea unei secvențe = numărul de elemente care o formează
- suma unei secvențe = suma elementelor care o formează
- secvența nebanală = secvența de lungime cel puțin egală cu **2**
- **N**-secvență = secvență a cărei sumă este divizibilă cu **N** (secvența poate fi și banală)
- **N**-secvență nebanală = secvență nebanală a cărei sumă este divizibilă cu **N**.

Cerințe

Scrieți un program care să citească numărul natural **N** și apoi șirul de **N** elemente. Programul determină:

1. numărul de **N**-secvențe nebanale din șir;
2. cea mai mare lungime a unei **N**-secvențe din șir;
3. cea mai mare sumă a unei **N**-secvențe din șir.

Date de intrare

Fișierul de intrare `secvente.in` conține pe prima linie două numere naturale **C** și **N**, separate printr-un singur spațiu, **C** reprezentând cerința care trebuie rezolvată (**1**, **2** sau **3**). A doua linie a fișierului conține cele **N** elemente numere întregi, separate prin câte un spațiu.

Date de ieșire

Fişierul de ieşire `secvente.out` va conţine:

- dacă $\mathbf{C = 1}$, pe prima linie, un număr natural reprezentând numărul de N-secvențe nebanale din șir (răspunsul la cerinţa **1**);
- dacă $\mathbf{C = 2}$, pe prima linie, un număr natural reprezentând cea mai mare lungime a unei N-secvențe din șir (răspunsul la cerința **2**);
- dacă $\mathbf{C = 3}$, pe prima linie, un număr natural reprezentând cea mai mare sumă a unei N-secvențe din șir (răspunsul la cerința **3**).

Restricții și precizări

- **N** număr natural, $\mathbf{2 \leq N \leq 100\ 000}$
- elementele șirului sunt numere întregi din interval închis $[\mathbf{-10^9}, \mathbf{10^9}]$
- în șirul de numere dat există cel puțin o **N**-secvență a cărei sumă este un număr natural
- numărul întreg negativ **X** este divizibil cu numărul natural nenul **N** dacă restul împărțirii modulului lui **X** la **N** este **0** (de exemplu, $\mathbf{X = -30}$ este divizibil cu $\mathbf{N = 6}$, iar $\mathbf{X = -45}$ nu este divizibil cu $\mathbf{N = 6}$).
- Pentru pentru fiecare dintre cerințele 1 și 2 se acordă $30p$, iar pentru cerința 3 se acordă $40p$.

Exemple

1)	secvente.in	secvente.out	**Explicații**
	1 10 -9 -3 4 -10 -1 -16 18 18 -10 50	8	Se rezolvă cerința **1**. Șirul are **N** = **10** elemente întregi: -9, -3, 4, -10, -1, -16, 18, 18, -10, 50. În *Figura*1 sunt marcate cele 8 N-secvențe nebanale.
2)	secvente.in	secvente.out	**Explicații**
	2 10 -9 -3 4 -10 -1 -16 18 18 -10 50	9	Se rezolvă cerința **2**. Șirul are **N** = **10** elemente întregi: -9, -3, 4, -10, -1, -16, 18, 18, -10, 50 Cea mai lungă dintre aceste secvențe este **N**-secvența: -3, 4, -10, -1, -16, 18, 18, -10, 50. Lugimea acestei **N**-secvențe este 9.
3)	secvente.in	secvente.out	**Explicații**
	3 10 -9 -3 4 -10 -1 -16 18 18 -10 50	60	Se rezolvă cerința **3**. Șirul are **N** = **10** elemente întregi: -9, -3, 4, -10, -1, -16, 18, 18, -10, 50 Suma maximă a unei **N**-secvențe este 60 (suma **N**-secvenței: -16, 18, 18, -10, 50).

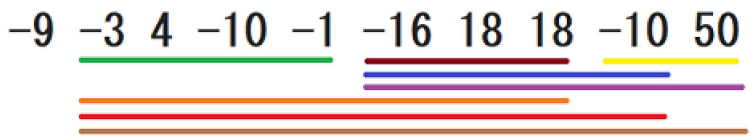


Figura 1

Descrierea soluției

O soluție, cu complexitate $\mathcal{O}(\mathbf{N})$ se poate obține astfel: odată cu citirea elementelor șirului $\mathbf{v[1]}, \mathbf{v[2]}, ..., \mathbf{v[N]}$ se construiesc sumele parțiale $\mathbf{S[k]} = \mathbf{v[1]} + \mathbf{v[2]} + ... + \mathbf{v[k]} = \mathbf{S[i-1]} + \mathbf{v[k]}$. Se va utiliza un vector de frecvență **Rest**[] a valorilor resturilor împărțirilor acestor sume parțiale la **N**. La fiecare rest negativ se va aduna **N**, rezultând un rest pozitiv din mulțimea $\{0, 1, 2, \ldots, \mathbf{N}-1\}$.

Cerința 1 (30 puncte)

Se observă că orice N-secvenţă se poate construi cu elementele aflate între oricare două poziţii cu sumele cu acelaşi rest. Numărul de secvențe pentru un rest dat este egal cu numărul de perechi pe care îl putem forma cu pozițiile care au dat sume având acel rest (dacă avem **x** poziții, numărul de perechi este $\mathbf{x(x-1)/2}$). Facem suma acestor valori pentru fiecare rest iar din această sumă scădem numărul de secvențe banale care este egal cu numărul de valori din șirul v care sunt multiplii ai numărului **N**.

Cerința 2 (30 puncte)

Pentru fiecare valoare distinctă a resturilor se va memora poziția p_1 a primei apariții a unei sume cu acest rest și poziția p_2 a ultimei apariții a unei sume cu acest rest. Astfel, numerele din șir situate între p_1 (exclusiv) și p_2 (inclusiv) formează o N-secvență de lungime $p_2 - p_1$ care este maximală relativ la valoarea restului respectiv. Căutăm apoi maximul dintre aceste lungimi.

Cerința 3 (40 puncte)

În timp ce se construiesc cei doi vectori (vectorul sumelor parțiale și vectorul frecvențelor de apariție) se păstrează pentru fiecare rest posibil poziția unde începe o potențială secvență de sumă maximă. Pentru fiecare element din șirul dat se calculează suma maximă a secvenței care se termină în el si se compară cu o variabilă în care memorăm suma maximă (inițializată cu 0 deoarece se garantează că există cel puțin o **N**-secvență de sumă pozitivă).

Pentru fiecare element $\mathbf{v}[\mathbf{k}]$ din șir ($k = 1, 2, ..., \mathbf{N}$) vom proceda astfel:

- se calculează suma parțială $\mathbf{S}[\mathbf{k}] = \mathbf{S}[\mathbf{k-1}] + \mathbf{v}[\mathbf{k}]$;
- se calculează restul $\mathbf{r} = \mathbf{S}[\mathbf{k}] \ \% \ \mathbf{N}$;
- dacă $\mathbf{r} < \mathbf{0}$ atunci $\mathbf{r} = \mathbf{r} + \mathbf{N}$;
- $\mathbf{Rest}[\mathbf{r}] = \mathbf{Rest}[\mathbf{r}] + \mathbf{1}$;
- dacă $\mathbf{v}[\mathbf{k}] \ \% \ \mathbf{N} = \mathbf{0}$ atunci $\mathbf{Smax} = \mathbf{max}(\mathbf{Smax}, \mathbf{v}[\mathbf{k}])$;
- dacă $\mathbf{S}[\mathbf{k}] \ \% \ \mathbf{N} = \mathbf{0}$ atunci $\mathbf{Smax} = \mathbf{max}(\mathbf{Smax}, \mathbf{S}[\mathbf{k}]$;
- dacă $\mathbf{Rest}[\mathbf{r}] = \mathbf{1}$ (adică, **r** apare pentru prima dată) atunci $\mathbf{SumaRest}[\mathbf{r}] = \mathbf{s}[\mathbf{k}]$
- altfel
  - dacă $\mathbf{S}[\mathbf{k}] - \mathbf{SumaRest}[\mathbf{r}] < \mathbf{0}$ (adică suma **N**-secvenței curente este negativă), atunci $\mathbf{SumaRest}[\mathbf{r}] = \mathbf{S}[\mathbf{k}]$;
  - $\mathbf{Smax} = \mathbf{max}(\mathbf{Smax}, \mathbf{S}[\mathbf{k}] - \mathbf{SumaRest}[\mathbf{k}])$

La final, se va afișa **Smax**.

4.3 Problema Valoare

Propunător: prof. Veronica Raluca Costineanu

- Colegiul Național "Ștefan cel Mare", Suceava

Enunț

Valoarea unei litere este dată de numărul ei de ordine în alfabet (**A** are valoarea **1**, **B** are valoarea **2**, **C** are valoarea **3**, etc.).

Un *text special* este format din una sau mai multe litere mari ale alfabetului englez și nu conține alte tipuri de caractere.

Valoarea unui text special o definim ca fiind egală cu suma valorilor literelor care îl compun. De exemplu, textul special **ABAC** are valoarea **7** (7=1+2+1+3).

Un text special poate fi transcris într-o formă *restrânsă* prin utilizarea perechilor de paranteze și a numerelor naturale. De exemplu, textul (**ABAC**)**2E2**(**CD**)**2E** reprezintă transcrierea în forma restrânsă a textului **ABACABACEECDCDE** deoarece secvența **ABAC** apare de două ori consecutiv, litera **E** are două apariții consecutive și la fel secvența de litere **CD**.

Transcrierea în formă restrânsă a unui text special se realizează prin înlocuirea fiecărei secvențe care are apariții consecutive în text cu secvența scrisă între o pereche de paranteze rotunde urmată apoi de numărul de repetiții. De exemplu, textul special **ABABAB** se transcrie în forma restrânsă în (**AB**)**3**.

Dacă, după transcriere, textul în formă restrânsă conține secvențe care au apariții consecutive în text atunci și acesta se va transcrie în formă restrânsă. De exemplu, textul special **ABABCABABC** se poate transcrie întâi în forma restrânsă în (**ABABC**)**2** sau în (**AB**)**2C**(**AB**)**2C** iar apoi în ((**AB**)**2C**)**2**.

Dacă textul special nu conține nicio secvență cu apariții consecutive în text, atunci forma restrânsă a textului este identică cu forma inițială. De exemplu, textul special **ABAC** are forma restrânsă **ABAC** care este identică cu acest text special.

Cerințe

Scrieți un program care citește un șir de caractere S reprezentând forma restrânsă a unui text special, și apoi determină:

1. numărul literelor distincte care apar în textul special;
2. suma numerelor care apar în forma restrânsă a textului special;
3. valoarea textului special dat în forma restrânsă.

Date de intrare

Fișierul de intrare `valoare.in` conține:

- pe prima linie, un număr natural **C** (1, 2 sau 3) reprezentând cerința care trebuie rezolvată;
- pe a doua linie, șirul de caractere **S** reprezentând forma restrânsă a unui text special.

Date de ieșire

Fişierul de ieşire `valoare.out` va conţine:

- dacă $\mathbf{C = 1}$, pe prima linie, un număr natural reprezentând numărul literelor distincte care apar în textul special (răspunsul la cerinţa **1**);
- dacă $\mathbf{C = 2}$, pe prima linie, un număr natural reprezentând suma numerelor care apar în forma restrânsă a textului special (răspunsul la cerința **2**);
- dacă $\mathbf{C = 3}$, pe prima linie, un număr natural reprezentând valoarea textului special dat în forma restrânsă (răspunsul la cerința **3**).

Restricții și precizări

- Șirul de caractere **S** ce reprezintă forma restrânsă a unui text special poate avea cel mult **1000** de caractere (litere mari ale alfabetului englez, perechi de paranteze rotunde, cifre).
- Numerele naturale care apar în forma restrânsă a unui text special sunt nenule și aparțin intervalului închis $[\mathbf{2}, \mathbf{9999}]$.
- Valoarea unui text special este un număr natural format din cel mult **12** cifre.
- Pentru cerința 1 se acordă $10p$, pentru cerința 2 se acordă $20p$, iar pentru cerința 3 se acordă $70p$.
- Pentru teste în valoare de $10p$, formă restrânsă a textului special este identică cu textul special și cerința este 3.

Exemple

1)	`valoare.in`	`valoare.out`	**Explicații**
	`1` `(ABAC)2E2(CD)2E`	5	Se rezolvă cerința **1**. Forma restrânsă a textului special este `(ABAC)2E2(CD)2E`. În textul special apar doar literele A, B, C, D și E.
2)	**`valoare.in`**	**`valoare.out`**	**Explicații**
	`2` `(ABAC)2E2(CD)2E`	6	Se rezolvă cerința **2**. Forma restrânsă a textului special este `(ABAC)2E2(CD)2E`. În textul special apare doar numărul 2, de 3 ori. Astfel suma este 6 (=2+2+2).
3)	**`valoare.in`**	**`valoare.out`**	**Explicații**
	`3` `(ABAC)2E2(CD)2E`	43	Se rezolvă cerința **3**. Forma restrânsă `(ABAC)2E2(CD)2E` reprezintă transcrierea textului special `ABACABACEECDCDE`. În textul special apar 4 de A, 2 de B, 4 de C, 2 de D și 3 de E. Astfel, valoarea textului special este egală cu 43. (43=4*1+2*2+4*3+2*4+3*5).

Descrierea soluției

O soluție se poate obține astfel:

Cerința 1 (10 puncte)

- folosim un vector caracteristic pentru a identifica literele distincte care apar în textul special.
- Complexitate $\mathcal{O}(n)$, unde n este lungimea textului

Cerința 2 (20 puncte)

- se parcurge șirul pentru a identifica secvențele formate doar din cifre, fiecare secvență se transformă în valoare numerică și se adună la suma generală
- Complexitate $\mathcal{O}(n)$, unde n este lungimea textului

Cerința 3 (70 puncte)

- folosim o stivă (simulată cu un vector) în care vom adăuga pe rând elemente după cum urmează:

  ◦ dacă în text urmează o paranteză deschisă adăugăm pe stivă un element care să marcheze acest lucru, de exemplu o valoare negativă (pentru a ști că aici începe repetiția unei secvențe)

  ◦ dacă în text urmează o literă atunci adăugăm pe stivă valoarea ei

  ◦ dacă urmează o paranteză închisă atunci sumăm și eliminăm elementele de pe stivă, până la prima paranteză deschisă pe care o întâlnim, și înlocuim paranteza cu valoarea sumei calculate (pentru a păstra pe stivă valoarea pe care o are textul cuprins între paranteze)

  ◦ dacă întâlnim un număr atunci valoarea din vârful stivei se înmulțește cu numărul și este înlocuită de acest produs (valoarea din vârful stivei reprezintă valoarea secvenței de text care trebuie repetat de număr ori)

- valoarea textului este dată de suma elementelor rămase la sfârșit pe stivă.
- Complexitate $\mathcal{O}(n)$, unde n este lungimea textului

Soluție alternativă - Cerința 3 - propusă de prof. Stelian Ciurea (ULB Sibiu)

- folosim o funcție recursivă în care parsăm șirul:
  - caracterele alfanumerice le tratăm în același mod ca în descrierea precedentă;
  - când întâlnim o paranteză deschisă apelăm recursiv funcția;
  - când întâlnim o paranteză închisă returnăm valoarea calculată în funcție.
  - rezultatul returnat de primul apel al acestei funcții reprezintă rezultatul problemei. Ideea de rezolvare este similară celei din descrierea precedentă cu singura diferență că folosim stiva sistemului prin mecanismul de apel și revenire în cazul funcțiilor recursive și nu o stivă implementată efectiv în program.
- folosim funcții de eliminare și inserare în șir (foarte simplu de utilizat dacă folosim string-uri) după cum urmează:
  - determinăm poziția primei ”)” din șir;
  - apoi parcurgem șirul în sens invers și determinăm poziția parantezei ”(” pereche;
  - construim un subșir cu elementele din șir aflate între aceste două poziții;
  - evaluăm acest subșir (care deci nu conține decât litere și cifre);
  - înmulțim rezultatul obținut cu valoarea numărului aflat după ”)” ;
  - stergem din șirul inițial subșirul evaluat (cel cuprins între paranteze și numărul aflat după ”)”;
  - construim un șir format din litera A (deoarece A are valoarea 1) urmată de valoarea obținută în pasul precedent;

- inserăm acest șir în poziția de unde am efectuat eliminarea;
- reluăm acești pași până am eliminat toate parantezele din șir;
- evaluăm șirul astfel obținut care nu mai conține decât litere și cifre;
- algoritmul se incadrează în timp pentru toate testele chiar dacă parcurge în mod repetat șirul dat.
- pentru a ilustra algoritmul arătăm transformările prin care trece șirul dat ca exemplu în enunțul problemei:

  (**ABAC**)**2E2**(**CD**)**2E**: prima pereche de paranteze este (**ABAC**), calculăm $\mathbf{A}+\mathbf{B}+\mathbf{A}+\mathbf{C}=\mathbf{7}$, înmulțim cu **2**, obținem **14**; eliminăm subșirul (**ABCD**)**2** și în locul lui inserăm **A14**; rezultă șirul

  A14E2(**CD**)**2E**: analog (**CD**): $\mathbf{C}+\mathbf{D}=\mathbf{7}$ înmulțim cu **2**, obținem **14**, eliminăm (**CD**)**2**, inserăm **A14**, rezultă șirul

  A14E2A14E : la acesta calculăm prin parsare $\mathbf{1}\cdot\mathbf{14}+\mathbf{5}\cdot\mathbf{2}+\mathbf{1}\cdot\mathbf{14}+\mathbf{5}=\mathbf{43}$

Capitolul 5

Baraj Juniori

5.1 Problema Intergalactic

Propunători: stud. Pop Ioan-Cristian, stud. Tulbă-Lecu Theodor-Gabriel

Universitatea Politehnică din București

Enunț

În secolul al XXIII-lea, oamenii au început să străbată spațiul intergalactic. Navele cu ajutorul cărora aceștia călatoresc sunt cu adevărat minuni ale tehnologiei, ele folosind un tip foarte exotic de combustibil.

Acest tip de combustibil se poate obține prin combinarea a exact doi reactanți, unul stabil cu unul instabil.

Fiecare reactant are atribuită o valoare sub forma unui număr natural nenul. Spunem despre un reactant că este stabil dacă valoarea acestuia este un număr prim și că este instabil dacă valoarea acestuia nu este număr prim.

Totuși, nu toate tipurile de combustibil sunt la fel de valoroase. După cum v-ați aștepta, prețul unui tip de combustibil este egal cu suma valorilor reactanților din care acesta este compus.

Cerință

Știind că pe piața intergalactică există N reactanți, să se răspundă la T întrebări de tipul: care este prețul celui de-al K-lea cel mai ieftin tip de combustibil care poate fi creat folosind doar reactanții disponibili pe piață.

Date de intrare

Fișierul *intergalactic.in* va conține pe prima linie numerele N și T, reprezentând numărul de reactanți de pe piață și numărul de întrebări.

Pe a doua linie se vor afla N numere separate prin câte un spațiu, reprezentând valorile celor N reactanți.

Pe următoarele T linii se va afla câte un singur număr K, cu semnificația din enunț.

Date de ieșire

Fișierul *intergalactic.out*, va conține T linii, pe linia i aflându-se un singur număr reprezentând răspunsul pentru întrebarea i.

Restricții și precizări

- $1 \leq T \leq 10$
- $1 \leq N \leq 200\ 000$
- $1 \leq K \leq 10^9$
- Mereu se vor putea crea cel putin K tipuri de combustibil.
- Nu vor exista 2 reactanți cu aceeași valoare.
- Pe piață vor exista cel puțin un reactant stabil și unul instabil.
- Valoarea unui reactant este mai mică sau egală cu 2 000 000.
- Două tipuri de combustibili cu acelasi preț sunt diferiți dacă provin din perechi de reactanți diferite.
- Pentru teste în valoare de 7 puncte: Pe piață există fie un singur reactant stabil, fie un singur reactant instabil.
- Pentru alte teste în valoare de 11 puncte: $K \leq 100$
- Pentru alte teste în valoare de 21 de puncte: $N \leq 1\ 000$
- Pentru alte teste în valoare de 28 de puncte: $N \leq 30\ 000$
- Pentru alte teste în valoare de 33 de puncte: nu există alte restricții

Exemple

	intergalactic.in	intergalactic.out	Explicații
	8 3 3 7 2 1 11 8 10 4 15 7 11	19 11 13	Reactanții instabili sunt: $1, 4, 8, 10$ Reactanții stabili sunt: $2, 3, 7, 11$ Valorile ordonate ale tipurilor de combustibil ce pot fi creați sunt: $3, 4, 6, 7, 8, 10, 11, 11, 12, 12, 13, 15, 15, 17, 19, 21$. Al 15-lea cel mai ieftin combustibil are valoarea 19. Al 7-lea cel mai ieftin combustibil are valoarea 11. Al 11-lea cel mai ieftin combustibil are valoarea 13.

Descrierea soluției

Soluție pentru cazul în valoare de 7 puncte, în care pe piață există fie un singur reactant stabil, fie un singur reactant instabil

Problema se reduce la identificarea unicului reactant (fie stabil, fie instabil), respectiv la sortarea crescătoare a celorlalți reactanți. Rezultatul va fi suma dintre reactantul unic și al k-lea reactant din vectorul sortat.

Pentru determinarea numerelor prime, se pot folosi atât ciurul lui Eratostene, cât și algoritmul de determinare în complexitate temporală $\mathcal{O}(\sqrt{N})$.

Soluție pentru cazul în valoare de 11 puncte, în care $K \leq 100$

Se observă că este nevoie strict de primii 100 de reactanți stabili, respectiv instabili (sortați în ordine crescatoare). Se construiesc sumele alcauite dintr-un reactant stabil și reactant instabil, iar apoi se sortează aceste sume. Astfel, rezultatul este a K-a suma.

Pentru determinarea numerelor prime, doar cirul lui Eratostene va obține punctaj maxim.

Soluție pentru cazul în valoare de 21 de puncte, în care $N \leq$1000

Se implementează aceeași solutie ca la cerința precedentă, cu precizarea că va trebui să procesam toți reactanții stabili, respectiv instabili.

Pentru determinarea numerelor prime, se pot folosi atât ciurul lui Eratostene, cât și algoritmul de determinare în complexitate temporală $\mathcal{O}(\sqrt{N})$.

Soluție pentru cazul în valoare de 28 de puncte, în care $N \leq 30000$

Aceste 28 de puncte se vor obtine pentru o solutie ineficienta, insa foarte asemanatoare cu soluția de 100 de puncte.

Determinăm numerele prime și construim doi vectori sortați: unul cu numere prime, unul cu numere neprime.

Observația cu care se pot obține cele 100 de puncte este aceea că soluția trebuie căutată binar. Fie valoarea propusă de cautarea binara (la un pas) X. Atunci, rămane să verificăm cate sume au valoarea strict mai mică decât X. Dacă numarul de sume este mai mic, atunci înseamnă că există o posibilitate ca X să fie soluția problemei, și căutăm o valoare mai mare. În caz contrar, înseamnă că soluția este mai mică decât X, deci căutăm o valoare mai mică. Soluția este ultima valoare propusă validă.

Pentru a calcula numarul de sume mai mici decât X, putem cauta binar, pentru fiecare număr prim (notăm cu A), care este cel mai mare număr neprim (notăm cu B) pentru care $A + B < X$. Având în vedere că vectorul este sortat, toate numerele din stânga lui B vor respecta și ele restricția (suma să fie mai mică decat X). Deci, vom aduna într-un contor, pentru fiecare valoare A, cate numere din celalalt vector pot fi alese.

O implementare a soluției de 100 de puncte, însă care determină în complexitate temporală $\mathcal{O}(\sqrt{N})$ numerele prime și nu cu ciurul lui Eratostene va obține aceste 28 de puncte.

Soluție oficială – 100 de puncte

Vom proceda la fel ca la soluția anterioară, însă schimbam metoda prin care determinăm câte sume sunt mai mici decat X. Plecăm de la observația că, trecând de la un numar prim la urmatorul, valoarea maximă a numarului neprim este cel mult egală cu valoarea maximă de la pasul precedent. Astfel, vom folosi un iterator prin vectorul de numere neprime. La trecerea la urmatorul număr prim, mutăm interatorul la stânga până când suma devine mai mică sau egală cu X (dacă este cazul).

O implementare a acestei soluții, însa fară ciurul lui Eratostene, nu va obține punctaj maxim.

5.2 Problema Cârtița

Propunători: prof.Daniela Lica
Centrul Județean de Excelență Prahova, Ploiești

Enunț

În grădina lui Macarie există un șir de N morcovi, numerotați de la 1 la N. Ca să știe unde sunt plantați, Macarie a făcut câte o grămăjoară de pământ în dreptul fiecărui morcov și a notat înălțimea fiecăreia exprimată în centimetri. Astfel morcovul i are în dreptul său o grămăjoară de pământ cu înălțimea de $h[i]$ centimetri.

O cârtiță neastâmpărată sapă galerii subterane pe sub morcovii lui Macarie. Când sapă o galerie către un morcov, tot pământul rezultat îl scoate afară modificând astfel înălțimea grămăjoarei corespunzătoare acelui morcov, dar și ale celorlalți morcovi din grădină.

Dacă în urma săpării unei galerii către morcovul de pe poziția pos înălțimea grămăjoarei lui a crescut cu x centimetri, atunci înălțimile grămăjoarelor tuturor morcovilor se modifică după următoarea regulă ce depinde de un număr K:

- înălțimea grămăjoarei morcovului $pos-1$ se modifică cu $x-K$ centimetri iar a morcovului $pos+1$ cu $x+K$ centimetri,
- înălțimile gramăjoarelor morcovilor $pos-2$ și $pos+2$ se modifică cu $x-2\cdot K$ respectiv $x+2\cdot K$ centimetri
- În caz general, înălțimea se modifică după următoarele reguli pentru fiecare morcov din grădină:

  Înălțimea $h[pos-i]$ devine $h[pos-i]+x-i\cdot K$, pentru fiecare i, $1\leq i\leq pos-1$

  Înălțimea $h[pos+i]$ devine $h[pos+i]+x+i\cdot K$, pentru fiecare i, $1\leq i\leq N-pos$

Cerință

Se cunosc înălțimile inițiale ale tuturor celor N grămăjoare și cele U modificări făcute de cârtiță asupra înălțimilor grămăjoarelor de pământ ale morcovilor.

Știm că în cadrul unei secvențe continue de morcovi cel mai tentant pentru cârtiță este morcovul cu cea mai mică înălțime a grămăjoarei de pământ.

Ajutați-l pe Macarie să identifice înălțimea grămăjoarei celui mai tentant morcov, pentru mai multe intervale date, după efectuarea tuturor modificărilor realizate de cârtiță.

Date de intrare

Fişierul de intrare *cartita.in* conţine pe prima linie un număr natural N având semnificația din enunț.

Pe următoarea linie se află N numere naturale despărțite prin câte un spațiu, reprezentând în ordine, înălțimile grămăjoarelor de pământ, al i-lea număr reprezentând înălțimea inițială $h[i]$, a grămăjoarei din dreptul morcovului i.

Pe a treia linie se află un număr natural U reprezentând numărul de morcovi către care cărtița a săpat galerii.

Pe următoarele U linii se află câte un triplet de numere naturale pos, x, K, separate între ele prin câte un spațiu, cu semnificația că în urma săpării unei galerii către morcovul pos înălțimea grămăjoarei lui a crescut cu x centimetri, iar celelalte înălțimi se modifică, după regula descrisă în enunț.

Pe următoarea linie se găsește numărul Q reprezentând numărul de intervale unde se dorește identificarea înălțimii minime a unei grămăjoare corespunzătoare celui mai tentant morcov.

Pe următoarele Q linii sunt câte două numere naturale L, R, separate între ele printr-un spațiu, reprezentând capetele intervalului de morcovi investigat.

Date de ieșire

Fişierul de ieşire *cartita.out* va conţine Q linii, pe fiecare găsindu-se, în ordine, răspunsul la intervalele investigate.

Restricții și precizări

- $1 \leq N \leq 100\ 000$
- $1 \leq U \leq 300\ 000$
- $1 \leq Q \leq 200\ 000$
- $1 \leq L \leq R \leq N$
- Inițial $1 \leq$ h[i] $\leq N$, pentru fiecare i, $1 \leq i \leq N$
- $1 \leq$ pos $\leq N$, $1 \leq$ x ≤ 400 și $-21 \leq$ K ≤ 21, $K \neq 0$
- **Subtask 1 – 12 puncte**: $N, U, Q \leq 2\ 000$
- **Subtask 2 – 9 puncte**: $N, U \leq 2\ 000$ și $Q \leq 200\ 000$
- **Subtask 3 – 25 de puncte**: $N, U \leq 75\ 000$ și $Q \leq 15$
- **Subtask 4 – 17 puncte**: $N, U, Q \leq 75\ 000$
- **Subtask 5 – 16 puncte**: $N \leq 100\ 000$, $U \leq 300\ 000$ și $Q \leq 15$
- **Subtask 6 – 7 puncte**: $N \leq 100\ 000$, $U \leq 300\ 000$ și $Q \leq 10\ 000$
- **Subtask 7 – 14 puncte**: Nu există restricții suplimentare.

Exemple

	cartita.in	cartita.out	Explicații
	6 1 6 2 4 3 4 3 4 5 2 2 4 1 4 2 8 3 1 6 2 4 4 4	-19 3 -17	După prima galerie săpată către morcovul 4, șirul înălțimilor celor $N = 6$ grămăjoare a devenit: $(0, 7, 5, 9, 10, 13)$. După a doua galerie săpată către morcovul 2, șirul înălțimilor celor $N = 6$ grămăjoare a devenit: $(3, 11, 10, 15, 17, 21)$. După a treia galerie săpată către morcovul 4, șirul înălțimilor celor $N = 6$ grămăjoare a devenit: $(-19, -3, 4, 17, 27, 39)$ Pe intervalul $[1, 6]$, înălțimea minimă este -19, pe intervalul $[2, 4]$, înălțimea minimă este -3, iar pentru ultimul interval investigat $[4, 4]$ înălțimea minimă este 17.

Descrierea soluției

Problema de față constă în prelucrarea eficientă a celor U *update*-uri asupra întregului șir de valori, și a celor Q *query*-uri de aflare a minimului dintr-un interval dat.

Subtask 1 (12 puncte):

Fiecare *update* va fi efectuat parcurgând întreg șirul de N valori și modificând corespunzător înălțimile.

Fiecare *query* va fi efectuat parcurgând întreg șirul de N valori și reținând înălțimea minimă întâlnită.

Complexitatea de timp totală: $\mathcal{O}(U \cdot N + Q \cdot N)$.

Subtask-urile 1 și 2 (21 de puncte):

Fiecare *update* va fi efectuat parcurgând întreg șirul de N valori și modificând corespunzător înălțimile.

Pentru a răspunde corect și eficient la cele Q *query*-uri, se pot folosi următoarele tehnici și metode de aflare a minimului dintr-un interval compact: *Arbori de Intervale*, *Range Minimum Query*, respectiv *Șmenul lui Batog*.

Complexitatea de timp totală: $\mathcal{O}(U \cdot N + Q \cdot log_2 N)$, sau $\mathcal{O}(U \cdot N + N \cdot log 2N + Q)$, respectiv $\mathcal{O}(U \cdot N + Q \cdot \sqrt{N})$.

Subtask-urile 1, 2, 3 și 4 (63 de puncte):

Pentru a aborda eficient operațiile tip *update*, vom folosi tehnica https://cp-algorithms.com/data_structures/sqrt_decomposition.html *SqrtDecomposition* sau https://www.infoarena.ro/multe-smenuri-de-programare-in-cc-si-nu-numai *Șmenul lui Batog*. Astfel, fiecare *update* va fi prelucrat într-o manieră de tip *lazy*, modificând valoarea a cel mult $2 \cdot \sqrt{N}$ elemente propriu-zise din șirul dat, în complexitatea de timp $\mathcal{O}(\sqrt{N})$.

Mai exact, vom împărți șirul de N elemente în *bucket-uri* (compartimente, subsecvențe) compacte de mărime $\lfloor \sqrt{N} \rfloor$. Așadar, vom avea cel mult $(\sqrt{N} \pm 2)$ *bucket-uri*. Dacă notăm cu $DIM = \lfloor \sqrt{N} \rfloor$, împărțirea celor N indici după metoda descrisă mai sus va fi de forma:

$$[1, DIM][DIM + 1, 2 \cdot DIM] \ldots [(\lfloor \frac{N + DIM - 1}{DIM} \rfloor - 1) \cdot DIM + 1, N].$$

Pentru un *update* de forma: i x k, înseamnă că valorile elementelor din fiecare poziție pos ($1 \leq pos \leq N$) se vor modifica cu $modif_{pos} = (x + (pos - i) \cdot k)$; dacă $modif_{pos} \geq 0$, putem spune că valoarea elementului de la poziția pos crește, altfel aceasta scade. Poziția i se află în *bucket*-ul cu indicele (indexat de la 1): $y = \lfloor \frac{i+DIM-1}{DIM} \rfloor$. În cadrul acestui *bucket*, cu indicele y, vom efectua modificările într-o manieră de tip *brute-force*, parcurgând cel mult DIM valori. Pentru acele *bucket*-uri care sunt indexate cu o valoare $> y$, vom modifica doar valoarea primului element (de indice minim) din cadrul *bucket*-ului; pentru cele care sunt indexate cu o valoare $< y$, vom modifica doar valoarea ultimului element (de indice maxim) din cadrul *bucket*-ului.

După ce am prelucrat *lazy* cele U *update*-uri, dorim ca în $\mathcal{O}(N)$ operații să modificăm întregul șir inițial, pentru a reține exact valorile reale care rezultă din urma *update*-urilor. Pentru a actualiza corect valorile din șir, vom reține suma tuturor k-urilor care apar în cele U *update*-uri. Pe baza acestei sume și a valorilor modificate din capetele *bucket*-urilor, putem deduce corect și restul valorilor.

Pentru procesarea eficientă *online*, în $\mathcal{O}(1)$, a fiecărui *query*, vom alege să efectuăm o precalculare (în $\mathcal{O}(N \cdot log_2 N)$), în cadrul căreia vom folosi tehnica https://infoarena.ro/problema/rmq *Range Minimum Query*. De asemenea, soluții ce folosesc tehnicile *Arborilor de Intervale* și *Range Minimum Query* vor fi destul de eficiente pentru a satisface task-urile amintite mai sus.

O soluție ce tratează în $\mathcal{O}(U \cdot \sqrt{N})$ *update*-urile și într-o manieră *brute query*-urile ($\mathcal{O}(Q \cdot N)$), poate obține $12 + 25 = 37$ de puncte (Subtask-urile 1 și 3).

Prin urmare, complexitățile de timp dorite pentru a obține punctajul maxim, de 63 de puncte în acest caz, sunt: $\mathcal{O}(U \cdot \sqrt{N} + N \cdot log_2 N + Q)$, $\mathcal{O}(U \cdot \sqrt{N} + N + Q \cdot log_2 N)$, sau $\mathcal{O}(U \cdot \sqrt{N} + Q \cdot \sqrt{N})$.

Subtask-urile 5, 6, 7 și Soluția de 100 de puncte:

Se dorește ca prelucrarea celor U *update*-uri și modificarea corectă a șirului inițial de valori să se realizeze în complexitatea de timp totală: $\mathcal{O}(U + N)$.

Vom utiliza 3 variabile, după cum urmează: Sum_X = suma tuturor x care apar în cele U *update*-uri; Sum_K = suma tuturor K care apar în cele U *update*-uri; $Diff$ = suma tuturor termenilor de forma $i \cdot K$ care apar în cele U *update*-uri.

Ne putem imagina că toate cele U *update*-uri ar avea $i = 1$, și, astfel, putem modifica în $\mathcal{O}(1)$ fiecare înălțime dintre cele N, folosind o metodă similară cu $SumeParțiale$ sau $Șmenului lui Mars$: $final_i = initial_i + Sum_X + (i \cdot Sum_K - Diff)$, pentru fiecare înățime i, cu $1 \leq i \leq N$.

Pentru tratarea eficientă a celor U *update*-uri, în $\mathcal{O}(1)$ fiecare, și aplicarea metodei *brute* pentru cele Q *query*-uri ($\mathcal{O}(N \cdot Q)$), se poate satisface Subtask-ul 5. Îmbunătățind prelucrarea query-urilor folosind *Șmenul lui Batog*, se poate trece și Subtask-ul 6.

Pentru 100 de puncte, în plus față de cele descrise anterior, se vor folosi *Arbori de Intervale* sau *Range Minimum Query* pentru cele Q query-uri.

5.3 Problema Inno

Propunători: : prof. Dan Pracsiu
Liceul Teoretic Emil Racoviță Vaslui

Enunț

Se dau numerele naturale N și K, precum și un șir $a[1], a[2], \ldots, a[N]$ de numere naturale nenule. Din șir se poate elimina o singură secvență (eventual vidă) $a[i], a[i+1], ..., a[j]$ astfel că în șir rămân elementele $a[1], a[2]$, $\ldots, a[i-1], a[j+1]$, $\ldots, a[N]$. De exemplu, din șirul $a = [1, 2, 3, 4, 5, 7]$ se poate elimina secvența $3, 4, 5$ și rămâne $1, 2, 7$; sau se poate elimina secvența vidă și rămâne șirul inițial $1, 2, 3, 4, 5, 7$; sau se poate elimina $1, 2, 3, 4$ și rămâne șirul $5, 7$.

După eliminarea secvenței, elementele rămase formează un șir *inno* dacă aplicându-se operația `&` pe biți asupra lor, rezultatul este un număr care are cel puțin K biți de 1 în baza 2. De exemplu, dacă $a = (1, 2, 3, 4, 5, 7)$ și $K = 2$, atunci prin eliminarea secvenței $1, 2, 3, 4$ rămân elementele $5, 7$, iar $5\&7 = 5$, care are 2 biți de 1 în baza 2. Dar dacă se elimină secvența $3, 4, 5$ atunci rămân elementele $1, 2, 7$, iar $1\&2\&7 = 0$, deci nu este șir *inno*.

Cerință

Să se determine în câte moduri se poate elimina o secvență astfel încât elementele rămase să formeze șir *inno*.

Date de intrare

Fișierul de intrare *inno.in* conține pe prima linie numerele naturale N și K. Pe linia a doua se află N numere naturale reprezentând elementele șirului, separate prin câte un spațiu.

Date de ieșire

Fișierul de ieșire *inno.out* va conține pe prima linie un singur număr natural reprezentând numărul de moduri de a elimina o secvență astfel încât șirul rămas să fie *inno*.

Restricții și precizări

- $3 \leq N \leq 200\ 000$
- $1 \leq K \leq 29$
- $0 \leq a_i \leq 2^{31} - 1$
- `&` este operatorul de conjuncție pe biți. Dacă x și y sunt valori binare, atunci expresia $x\&y$ este egală cu 1 dacă și numai dacă $x = 1$ și $y = 1$. Deci $1\&1 = 1$, $0\&1 = 0$, $1\&0 = 0$, $0\&0 = 0$. Dacă a și b sunt numere naturale, atunci expresia $a\&b$ se efectuează la nivelul reprezentării în baza 2. De exemplu, dacă $a = 12$ și $b = 20$, atunci:

  $a\&b = 12\&20 = 01100_{(2)}\&10100_{(2)} = 00100_{(2)} = 4_{(10)}$
- Pentru teste în valoare de 12 puncte: $1 \leq K \leq 2$ și $3 \leq N \leq 25$
- Pentru alte teste în valoare de 23 de puncte: $500 \leq N \leq 8\ 000$
- Pentru alte teste în valoare de 65 de puncte: Nu există restricții suplimentare.

Exemple:

1)	inno.in	inno.out	Explicații
	4 2 10 7 5 15	5	Modalitățile sunt: • se elimină 10 și rămâne șirul 7, 5, 15, iar 7&5&15 = 5, care are 2 biți de 1 • se elimină 10,7 și rămâne șirul 5,15, iar 5&15 = 5, care are 2 biți de 1 • se elimină 10,7,5 și rămâne șirul 15, iar 15 are 4 biți de 1 • se elimină 7,5 și rămâne șirul 10,15, iar 10&15 = 10, care are 2 biți de 1 • se elimină 7,5,15 și rămâne șirul 10, iar 10 are 2 biți de 1
2)	**inno.in**	**inno.out**	**Explicații**
	5 4 7 7 6 1 62	1	Singura posibilitate este eliminarea secvenței 7,7,6,1. Rămâne doar numărul 62, care are 5 biți de 1.

Descrierea soluției

Soluție parțială – 12 puncte

Pentru fiecare pereche de indici (i, j) cu $1 \leq i \leq j \leq N$ se elimină secvența $a[i]$, $a[i+1]$, $\ldots, a[j]$ și se verifică dacă $a[1]\&a[2]\&\ldots\&a[i-1]\&a[j+1]\&\ldots\&a[N]$ are cel puțin K biți de 1. Complexitatea temporală a acestei soluții este $\mathcal{O}(N^3 \cdot log(N))$.

Soluție parțială – 35 de puncte

Construim vectorii st și dr de lungime N în care:

- $st[i] = a[1]\&a[2]\&\ldots\&a[i]$
- $dr[i] = a[i]\&a[i+1]\&\ldots\&a[N]$

Ca și la soluția anterioară, se verifică dacă în urma eliminării unei secvențe $a[i]$, $a[i+1]$, $\ldots, a[j]$, șirul rămas este un șir *inno*.

Trebuie să verificăm dacă $a[1]\&a[2]\&\ldots\&a[i-1]\&a[j+1]\&\ldots\&a[N]$ are cel puțin K biți de 1. Dar $a[1]\&a[2]\&\ldots\&a[i-1] = st[i-1]$, iar $a[j+1]\&\ldots\&a[N] = dr[j+1]$. Deci se testează dacă $st[i-1]\&dr[j+1]$ are cel puțin K biți de 1.

Complexitatea temporală va fi în acest caz $\mathcal{O}(N^2 \cdot log(N))$.

Soluție oficială – 100 de puncte

Ne vom folosi in continuare de vectorii st și dr definiți anterior:

Notăm cu:

- x = cea mai din dreapta poziție cu proprietatea că $st[x]$ are cel puțin K biți de 1 (dacă $a[1]$ are mai puțin de K biți de 1, atunci $x = 0$)
- y = cea mai din stânga poziție cu proprietatea că $dr[y]$ are cel puțin K biți de 1 (dacă $a[N]$ are mai puțin de K biți de 1, atunci $y = N + 1$)

Apar cazurile:

1. $x = N$, adică $st[n]$ are cel puțin N biți de 1, deci întregul șir este șir *inno*. În acest caz, orice secvență poate fi eliminată (mai puțin întreg șirul, dar se poate elimina în schimb secvența vidă), deci soluția problemei este în acest caz $N \cdot (N+1)/2$.
2. $x = 0$ și $y = N + 1$. În acest caz nu există nicio secvență
3. $1 \leq x < N$ și $y = N + 1$. În acest caz șirurile *inno* se obțin eliminând orice secvență de forma $a[i], a[i+1], \ldots, a[N]$, cu $i = 2, \ldots x + 1$, deci răspunsul la problemă este x.
4. $x = 0$ și $1 < y \leq N$. Atunci șirurile *inno* se obțin prin eliminarea secvențelor de forma $a[1], a[2], \ldots, a[i]$, cu $i = y - 1, \ldots, N - 1$, deci răspunsul la problemă este $N - y + 1$.
5. $x \geq 1$ și $y \leq n$. Din aceste condiții rezultă și că $x < y$, altfel ne-am afla în cazul 1. Deci în mod obligatoriu trebuie eliminată o secvență care conține pe $a[x+1], a[x+2], \ldots, a[y-1]$. Soluția problemei este dată de:
   a) eliminarea secvențelor de forma $a[1], a[2], \ldots, a[i]$, cu $i = y - 1, \ldots, N - 1$ ($N - y + 1$ soluții)
   b) pentru fiecare $i = 1, \ldots, x$, se caută cea mai din stânga poziție p, cu $p = y, \ldots, N$, cu proprietatea că $st[i]\&dr[p]$ are cel puțin K biți de 1; se elimină secvențe de forma $a[i+1], a[i+2], \ldots, a[j]$, cu $j = p - 1 \ldots, N$ ($N - j + 2$ soluții). Pentru fiecare $i = 1, \ldots, x$ determinarea poziției p se poate face fie prin căutare binară, fie prin tehnica *two pointers*.

Complexitatea algoritmului este $\mathcal{O}(NlogN)$, unde logaritmul se obține prin determinarea numărului de biți de 1.

Deși nu este necesar pentru obținerea punctajului maxim, putem îmbunătăți aceasta soluție în continuare.

Funcția de calculare a numărului de biți (dese ori întâlnita sub numbele de *popcount*), poate fi implementată în complexitate temporală $\mathcal{O}(1)$ în diferite moduri:

1. Folosind o tabelă de valori: Știind că numerele noastre sunt numere pe 32 de biți, putem să creăm o tabelă *cnt*, astfel încât $cnt[i]$ reprezintă numărul de biți de 1 ai numărului i, doar pentru numere de maxim 16 biți (numerele de la 0 la 65535). Această tabelă poate fi calculată cu ajutorul recurenței $cnt[i] = cnt[i/2] + (i\&1)$.

   Acum, pentru a calcula pentru orice număr x de 32 de biți câți biți de 1 are aprinși este suficient să calculăm câți biți sunt aprinși În prima jumătate a biților numărului și câți sunt aprinși în a doua jumătate. Pentru a realiza acest lucru ne vom folosi de tabela *cnt* astfel:

   $$\text{popcnt} = cnt[x >> 16] + cnt[x \ \texttt{\&}\ 65536]$$

2. Folosind funcții deja existente: în cazul compilatorului de C/C++ folosit în cadrul concursului – gcc/g++, această funcție se numește `__builtin_popcount`.
3. Implementând funcția *popcount* folosind operații pe biți. Puteți găsi mai multe detalii despre implementarea acestei funții optim aici:
   https : *//en.wikipedia.org/wiki/Hamming_weight*

Folosind o astfel de implementare, complexitatea temporală a soluției devine $\mathcal{O}(N)$.

Capitolul 6

Anexe - Exemple implementări soluții în limbajul C/C++

6.1 Soluții - Clasa a V-a

1. ktlon

```
/*
1.1. Autor: Florentina Ungureanu
*/
#include <bits/stdc++.h>
#define ull unsigned long long
using namespace std;
ifstream in("ktlon.in");
ofstream out("ktlon.out");
ull c,n,k,p,pr,mr,mf,sf,sr,M;
int main()
{
    in>>c>>n>>k;
    if(c==1)
    {
        for(int j=1; j<=k; ++j)
        {
            mf=mr=0;
            for(int i=1; i<=n; ++i)
            { in>>p;if(p>mf) mf=p;}
            for(int i=1; i<=n; ++i)
            { in>>p;if(p>mr) mr=p;}
            if(mf<mr) pr++;
        }
        out<<pr<<'\n';
    }
    else
    {        sr=sf=0;
        for(int j=1; j<=k; ++j)
        {
            mf=0;
            ull f[7]={0},r[7]={0};
            for(int i=1; i<=n; ++i)
            {
                in>>p;
                int ii=0;
                while(ii<5&&p<f[ii])ii++;
                for (int j=5; j>ii; j--)
                    f[j]=f[j-1];
                f[ii]=p;
            }
```

```
            for(int i=1; i<=n; ++i)
            {
                in>>p;
                int ii=0;
                while(ii<5&&p<r[ii])ii++;
                for (int j=5; j>ii; j--)
                    r[j]=r[j-1];
                r[ii]=p;
            }

            M=0;
            if(f[0]>r[0])
                while(M<5&&f[M]>r[0]){sf+=f[M]-r[M];M++;}
            else if(f[0]<r[0])
                while(M<5&&r[M]>f[0]){sr+=r[M]-f[M];M++;}
        }
        out<<max(sf,sr);
    }
    return 0;
}
```

```
/*
1.2. Autor: Georgeta Balacea
*/
#include<fstream>
#include<algorithm>
///#include<cmath>
using namespace std;
ifstream f("ktlon.in");
ofstream  g("ktlon.out");
unsigned long long
C,n,k,F[10001],R[10001],i,j,maxf,maxr,nrsf,nrsr,ok,x,SF,SR,nrpr,M;
int main()
{   f>>C>>n>>k;
    if(C==1)
    {   for(int j=1; j<=k; j++)
        {
            maxf=0;
            for(int i=1; i<=n; i++)
            {
                f>>x;
                maxf=max(maxf,x);
            }
            ok=0;
            for(int i=1; i<=n ; i++)
            {
                f>>x;
                if(x>maxf) ok=1;
            }
            nrpr+=ok;
        }
        g<<nrpr<<endl;
    }
```

```
    else
    {
        for(int j=1; j<=k; j++)
        {
            for(int i=1; i<=n; i++) f>>F[i];
            sort(F+1,F+n+1);
            for(int i=1; i<=n; i++) f>>R[i];
            sort(R+1,R+n+1);
            M=0;
            SF=0;
            SR=0;
            if(F[n]>R[n])
            {
                int i=n;
                while(i>=1 && F[i]>R[n])
                {
                    M++;
                    SF+=F[i];
                    SR+=R[i];
                    i--;
                }
                nrsf+=SF-SR;
            }
            else if(F[n]<R[n])
            {
                int i=n;
                while(i>=1 && R[i]>F[n])
                {
                    M++;
                    SF+=F[i];
                    SR+=R[i];
                    i--;
                }
                nrsr+=SR-SF;
            }
        }
        g<<max(nrsf,nrsr)<<endl;
    }
    f.close();
    g.close();
    return 0;
}
```

```
/*
1.2. Autor: Grecea Violeta
*/
#include <bits/stdc++.h>
using namespace std;
ifstream fin("ktlon.in");
ofstream fout("ktlon.out");
unsigned long long n, k, x, F[10],R[10],i,j,maxF,ct,proba,cerinta,
p,nrsteleF,M,s, nrsteleR;
bool ok;
int main()
{fin>>cerinta;
 if(cerinta==1)
     {fin>>n>>k;
     for(proba=1;proba<=k;proba++)
         { ok=0;  maxF=0;
         for(i=1;i<=n;i++)
              {fin>>x;
               if(x>maxF) maxF=x;
              }
         for(i=n+1;i<=2*n;i++)
              {fin>>x;
               if(x>maxF) ok=1;
              }
         if(ok==1) ct++;
         }
     fout<<ct<<'\n';
     }
 else
    {fin>>n>>k;
     if(n<5) p=n;
             else p=5;
     for(proba=1;proba<=k;proba++)
         {
          for(i=1;i<=n;i++)
              {fin>>x;
               j=p;
               while(j>=1 && x>F[j]) { F[j+1]=F[j];j--;}
               F[j+1]=x;
               }
         for(i=1;i<=n;i++)
              {fin>>x;  j=p;
               while(j>=1 && x>R[j]) { R[j+1]=R[j];j--;}
               R[j+1]=x;
               }
```

```
        M=0;
        if(F[1]>R[1])
                {j=1;
                 s=0;
                 while(j<=p && F[j]>R[1]) {s=s+F[j];M++;j++;}
                 for(j=1;j<=M;j++) s=s-R[j];
                 nrsteleF+=s;
                }
            else
              if(F[1]<R[1])
                {j=1;
                 s=0;
                 while(j<=p && R[j]>F[1]) {s=s+R[j];M++;j++;}
                 for(j=1;j<=M;j++) s=s-F[j];
                 nrsteleR+=s;
                }
         for(i=1;i<=p;i++) F[i]=0;
         for(i=1;i<=p;i++) R[i]=0;
         }
    if(nrsteleF>nrsteleR) fout<<nrsteleF<<'\n';
                else fout<<nrsteleR<<'\n';;
    }
return 0;
}
```

```
/*
1.2. Autor: Adriana Simulescu
*/
#include <fstream>
/// calcul cele mai mari 5 numere

using namespace std;

ifstream in("ktlon.in");
ofstream out("ktlon.out");

long long c, k, n, maxf, maxr, steleF, steleR, cf, cr, F[2005], R[2005];

int main()
{
 int i, j, x;
 in>>c>>k>>n;
```

```
 if(c==1)
    {
     for(i=1; i<=n; ++i)
     {
         maxf=0; maxr=0;
         for( j=1; j<=k; ++j)
         {
             in>>x;
             if(x>maxf) maxf=x;
         }
         for(j=1; j<=k; ++j)
         {
             in>>x;
             if(x>maxr) maxr=x;
         }
         if(maxf>maxr) cf++;
         else
            if(maxr>maxf)cr++;
     }
     out<<cr<<endl;
    }

else
    {
    for(i=1; i<=n; ++i)
     {
        for(j=1; j<=5; j++)
            F[j]=R[j]=-1;
        for( j=1; j<=k; ++j)
         {
             in>>x;
             if(x>F[1])
             {
                 F[5]=F[4]; F[4]=F[3]; F[3]=F[2]; F[2]=F[1]; F[1]=x;
             }
             else if(x>F[2])
             {
                 F[5]=F[4]; F[4]=F[3]; F[3]=F[2]; F[2]=x;
             } else if(x>F[3])
             {
                 F[5]=F[4]; F[4]=F[3]; F[3]=x;
             } else if(x>F[4])
             {
                 F[5]=F[4]; F[4]=x;
             }  else if(x>F[5])
             {
                 F[5]=x;
             }
         }
```

```
        for(j=1; j<=k; ++j)
        {
            in>>x;
            if(x>R[1])
            {
                R[5]=R[4]; R[4]=R[3]; R[3]=R[2]; R[2]=R[1]; R[1]=x;
            }
            else if(x>R[2])
            {
                R[5]=R[4]; R[4]=R[3]; R[3]=R[2]; R[2]=x;
            } else if(x>R[3])
            {
                R[5]=R[4]; R[4]=R[3]; R[3]=x;
            } else if(x>R[4])
            {
                R[5]=R[4]; R[4]=x;
            }  else if(x>R[5])
            {
                R[5]=x;
            }

        }
          if(F[1]>R[1])
           {
           int M=0;
           j=1;
           while(j<=5&&j<=k&&F[j]>R[1])
               {M++;  steleF=steleF+F[j]-R[j]; j++;}
           }



          else
          if(R[1]>F[1])
           {
            int M=0;
            j=1;
            while(j<=5&&j<=k&&F[1]<R[j])
               {M++;  steleR=steleR-F[j]+R[j]; j++;}
           }
     }
if(steleR>steleF)
     out<<steleR<<endl;
else out<<steleF<<endl;
    }
    return 0;
}
```

2. iepuraş

```
/*
2.1. Autor Georgeta Balacea
*/
#include<fstream>
using namespace std;
unsigned long long n,p,n1,n2,nrap;
int C,c,N,i,cerinta;
int cif[11];
unsigned long long nmax,nmin,sum;

ifstream f("iepuras.in");
ofstream g("iepuras.out");
int main()
{
    f>>cerinta;
    if(cerinta==2)
    {
        f>>N;
        for(i=1; i<=N; i++)
        {
            f>>n;
            C=n%9;
            if(C==0) C=9;
            nrap=0;
            p=1;
            n2=0;
            while(n>0)
            {
                c=n%10;
                n1=n/10;
                nrap+=n1*p;
                if(c>C) nrap+=p;
                else if(c==C) nrap+=n2+1;
                n2+=c*p;
                p*=10;
                n/=10;
            }
            g<<nrap<<'\n' ;
        }
    }
    else
    {   f>>N;
        for(i=1; i<=N; i++)
        {
            f>>n;nmin=0;nmax=0;
```

```
            for(int i=0; i<=9; i++) cif[i]=0;
            while(n)
            {
                cif[n%10]++;
                n/=10;
            }
            for(int i=9; i>=0; i--)
                if(cif[i]) nmax=nmax*10+i;
            int i=1;
            while(cif[i]==0) i++;
            cif[i]=0;
            nmin=i;
            for(i=0; i<=9; i++)
                if(cif[i]) nmin=nmin*10+i;
            sum=nmin+nmax;
            g<<sum<<endl;
        }


    }
    f.close();
    g.close();
    return 0;
}
```

```
/*
2.2. Autor Dan Spătărel
*/
#include <stdio.h>

int main() {
  freopen("iepuras.in", "r", stdin);
  freopen("iepuras.out", "w", stdout);
  int C, n;
  scanf("%d%d", &C, &n);
  for (int i = 0; i < n; i++) {
    long long egg;
    scanf("%lld", &egg);
    int checkSumDigit = egg % 9;
    if (checkSumDigit == 0) {
      checkSumDigit = 9;
    }
    if (C == 1) {
      bool isPresent[10];
      for (int d = 0; d <= 9; d++) {
        isPresent[d] = false;
      }
      while (egg > 0) {
        int digit = egg % 10;
        egg /= 10;
        isPresent[digit] = true;
      }
```

```
      long long maxNumber = 0;
      for (int d = 9; d >= 0; d--) {
        if (isPresent[d]) {
          maxNumber = maxNumber * 10 + d;
        }
      }
      int minDigit = 1;
      while (!isPresent[minDigit]) {
        minDigit++;
      }
      isPresent[minDigit] = false;
      long long minNumber = minDigit;
      for (int d = 0; d <= 9; d++) {
        if (isPresent[d]) {
          minNumber = minNumber * 10 + d;
        }
      }
      long long sum = minNumber + maxNumber;
      printf("%lld\n", sum);
    } else { // C == 2
      long long digitCounter = 0;
      long long power = 1;
      long long suffix = 0;
      while (egg > 0) {
        int digit = egg % 10;
        egg /= 10;
        if (digit == checkSumDigit) {
          digitCounter += egg * power + suffix + 1;
          //printf("+%lld\n", ai * power + suffix + 1);
        } else if (digit > checkSumDigit) {
          digitCounter += (egg + 1) * power;
          //printf("+%lld\n", (ai + 1) * power);
        } else { // digit < checkSumDigit
          digitCounter += egg * power;
          //printf("+%lld*%lld\n", ai, power);
        }
        suffix += digit * power;
        power *= 10;
      }
      printf("%lld\n", digitCounter);
    }
  }
  return 0;
}
```

```
/*
2.3. Autor: Grecea Violeta
*/
#include <fstream>
using namespace std;
ifstream fin("iepuras.in");
ofstream fout("iepuras.out");
int n, cerinta, i, cifcontrol, u,vf[10],c;
unsigned long long x,nr,p,maxim, minim, nrap;

int main()
{fin>>cerinta;
 if(cerinta==1)
    {fin>>n;
     for(i=1;i<=n;i++)
          {fin>>x;
           if(x==0) vf[0]++;
               else while(x>0) {vf[x%10]=1;
                                x=x/10;}
          for(c=9;c>=0;c--)
                if(vf[c]!=0) maxim=maxim*10+c;
         if(vf[0]!=0)
              for(c=1;c<=9;c++)
                   if(vf[c]!=0) {minim=c;vf[c]--;break;}
          for(c=0;c<=9;c++)
                if(vf[c]!=0) minim=minim*10+c;
           fout<<minim+maxim<<'\n';
        for(c=0;c<=9;c++) vf[c]=0;
        maxim=0;
        minim=0;
         }
    }
    else
     {fin>>n;
      for(i=1;i<=n;i++)
          {fin>>x;
           if(x%9==0) cifcontrol=9;
                 else cifcontrol=x%9;
            nr=0; nrap=0; p=1;
           while(x>0)
                {u=x%10;
                 nrap=nrap+(x/10)*p;
                 if(u>cifcontrol) nrap=nrap+p;
                    else if(u==cifcontrol) nrap=nrap+nr+1;
                nr=nr+u*p; p=p*10; x=x/10;
                }
            fout<<nrap<<'\n' ;
           }
        }
return 0;
}
```

```
/*
2.4. Autor: Adriana Simulescu
*/

#include <fstream>
#include <algorithm>
using namespace std;
ifstream in("iepuras.in");
ofstream out("iepuras.out");
int C,n, ap[10], s, cfc, j;
long long x;
 int c_control(long long x)
 {   int s;
     while(x>9)
     {
         s=0;
         while(x)
         {
             s+=x%10;
             x=x/10;
         }
         x=s;
     }
     return x;
 }

long long nr_cifre(long long k, int c)
{
     long long sol=0, p=1, n;
     n=k;
       while (k)
    {
      if (k % 10 > c)
            sol = sol + (k / 10 + 1) * p;
      if (k % 10 < c)
            sol = sol + (k / 10) * p;
      if (k % 10 == c)
            sol = sol + (k / 10) * p + n % p + 1;
      k /= 10;  p *= 10;
    }
return sol;
}
```

```
int main()
{
   in>>C>>n;
    if(C==2)
    {
         for( int i=1; i<=n; i++)
        {
            in>>x;
            int  cc=c_control(x);
            long long nr=nr_cifre(x, cc);
            out<<nr<<endl;
        }
    }
    else
        {
        for( int i=1; i<=n; ++i)
        {
            in>>x;
            for(int j=0; j<=9; ++j)
                ap[j]=0;
            do
            {
                ap[x%10]=1;
                x/=10;
            } while(x);
            long long y=0, z=0;
            for(int j=9; j>=0; --j)
                if(ap[j])
                    y=y*10+j;
            if(ap[0])
            {
                j=1;
                while(ap[j]==0)
                    j++;
                z=j;
                ap[j]=0;
            }
                for(j=0; j<=9; j++)
                      if(ap[j])
                        z=z*10+j;
           out<<y+z<<endl;
        }
        }
    return 0;
}
```

```
/*
2.5. Autor Dumitrascu Dan Octavian
*/
#include <fstream>
using namespace std;
ifstream f("iepuras.in");
ofstream g("iepuras.out");
long long C, n, i, x, mi, ma, k, aux, p, y, nn, nr, nre, j, uc, fr[15];
int main()
{
    f>>C;
    if (C==1)
    {
        f>>n;
        for (i=1;i<=n;i++)
        {
            f>>x;
            if (x==0) g<<0<<"\n";
            else
            {
            for (k=0;k<=9;k++)
                fr[k]=0;
            aux=x;
            while (aux>0)
            {
                fr[aux%10]=1;
                aux/=10;
            }
            ma=0;
            for (k=9;k>=0;k--)
                if (fr[k]==1) ma=ma*10+k;
            mi=0;
            for (k=1;k<=9;k++)
                if (fr[k]==1)
            {
                mi=k;
                fr[k]=0;
                break;
            }
            for (k=0;k<=9;k++)
                if (fr[k]==1) mi=mi*10+k;
            g<<mi+ma<<"\n";
            }
        }
    }
    else
    {
        f>>n;
        for (i=1;i<=n;i++)
        { f>>x;
          if (x==0) g<<1<<"\n";
          else
          {
            y=x%9;
```

```
            if (y==0) y=9;
            nre=0;
            aux=x;
            p=1;
            nn=0;
            while (aux>0)
            {

                uc=aux%10;
                if (uc>y) nre=nre+(aux/10+1)*p;
                else
                    if (uc==y) nre=nre+ aux/10*p+ nn+1;
                else nre=nre+aux/10*p;
                nn=nn+uc*p;
                p=p*10;
                aux=aux/10;
            }
            g<<nre<<"\n";
          }
        }
    }
    return 0;
}
```

3. tăieri

```
/*
  3.1. Autor Marius Nicoli
*/
#include <fstream>
using namespace std;
int F[9], G[9], f[9], i, n, m, x, j;
int main () {
    ifstream fin ("taieri.in");
    ofstream fout("taieri.out");
    fin>>n;
    for (i=1;i<=n;i++) {
        fin>>x;
        for (j=8;j>=1;j/=2) {
            f[j] += x/j;
            x%=j;
        }
    }
    fin>>m;
    while (m--) {
        for (i=8;i>=1;i/=2) {
            F[i] = f[i];
        }
```

```
        fin>>G[1]>>G[2]>>G[4]>>G[8];
        int ok = 1;
        for (i=8;i>=1;i/=2) {
            if (G[i] > F[i]) {
                ok = 0;
                break;
            }
            F[i/2] += (F[i]-G[i])*2;
        }

        fout<<ok;
        if (m!=0)
            fout<<" ";
        else
            fout<<"\n";
    }
    return 0;
}
```

```
/*
  3.2. Autor Dumitrascu Dan Octavian
*/

#include <fstream>
using namespace std;
ifstream f("taieri.in");
ofstream g("taieri.out");
int r, n, v[100005],bara[10], m, i, a,b,c,d,x1,x2,x4,x8;

int main()
{
    f>>n;
    for (i=1;i<=n;i++)
    {   f>>v[i];
        if (v[i]>=8)
        {
            bara[8]+=v[i]/8;
            r=v[i]%8;
            bara[r]++;
        }
        else
            if (v[i]>=4)
        {
            bara[4]+=v[i]/4;
            r=v[i]%4;
            bara[r]++;
        }
            else
                if (v[i]>=2)
            {
                bara[2]+=v[i]/2;
                r=v[i]%2;
                bara[r]++;
            }
            else
                if (v[i]==1)
                   bara[1]+=v[i];

    }
    bara[4]+=(bara[5]+bara[6]+bara[7]);
```

```
    bara[1]+=bara[5];
    bara[2]+=bara[6];
    bara[3]+=bara[7];
    bara[2]+=bara[3];
    bara[1]+=bara[3];
    x1=bara[1];
    x2=bara[2];
    x4=bara[4];
    x8=bara[8];
    f>>m;
    for (i=1;i<=m;i++)
    {   f>>a>>b>>c>>d;
        bara[1]=x1;
        bara[2]=x2;
        bara[4]=x4;
        bara[8]=x8;
        if (d>bara[8]) g<<0;
        else
        {
            bara[4]+=(bara[8]-d)*2;
            if (c>bara[4]) g<<0;
            else
            {
                bara[2]+=(bara[4]-c)*2;
                if (b>bara[2]) g<<0;
                else
                {   bara[1]+=(bara[2]-b)*2;
                    if (a>bara[1]) g<<0;
                    else g<<1;
                }
            }
        }
        if (i<=m) g<<" ";
    }
    return 0;
}
```

```
/*
3.3.Autor Georgeta Balacea
*/

#include<fstream>
#include<algorithm>

using namespace std;

int n,m,a,b,c,d,na,nb,nc,nd,v[100000],N[4],i,x,ok;

ifstream f("taieri.in");
ofstream g("taieri.out");

int main()
{
    f>>n;
    for(i=1;i<=n;i++)
    {   f>>v[i];
        x=v[i];
        N[3]+=x/8;
        x=x%8;
```

```
        N[2]+=x/4;
        x=x%4;
        N[1]+=x/2;
        x=x%2;
        N[0]+=x;
    }

    f>>m;

    for(i=1;i<=m;i++)
    {
        f>>a>>b>>c>>d;
        ok=1;
        na=N[0];nb=N[1];nc=N[2];nd=N[3];
        if(nd<d)ok=0;
        else
        {
            nc=nc+(nd-d)*2;
            if(nc<c) ok=0;
            else
                {nb=nb+(nc-c)*2;
                 if(nb<b) ok=0;
                 else
                 {
                     na=na+(nb-b)*2;
                     if(na<a) ok=0;
                 }
                }

        }
        g<<ok<<' ';
    }


    f.close();

    g.close();

    return 0;

}
```

```
/*
3.4.Autor Grecea Violeta
*/

#include <fstream>

using namespace std;
ifstream fin("taieri.in");
ofstream fout("taieri.out");
int n, k, nrb1, nrb2, nrb4, nrb8, a,b,c,d,i,lungime,ok, cnrb1, cnrb2, cnrb4,
cnrb8;

int main()
{fin>>n;
 for(i=1;i<=n;i++)
     {fin>>lungime;
      nrb8=nrb8+lungime/8;
      lungime=lungime-lungime/8*8;
```

```
      nrb4=nrb4+lungime/4;
      lungime=lungime-lungime/4*4;
      nrb2=nrb2+lungime/2;
      lungime=lungime-lungime/2*2;
      nrb1=nrb1+lungime;
     }
cnrb1=nrb1;
cnrb2=nrb2;
cnrb4=nrb4;
cnrb8=nrb8;

fin>>k;
for(i=1;i<=k;i++)
    {fin>>a>>b>>c>>d;
     ok=1;
     if(nrb8<d) {fout<<0<<" "; ok=0;}
          else {nrb4=nrb4+(nrb8-d)*2;
                if(nrb4<c) {fout<<0<<" "; ok=0;}
                     else {nrb2=nrb2+(nrb4-c)*2;
                           if(nrb2<b) {fout<<0<<" "; ok=0;}
                                 else {nrb1=nrb1+(nrb2-b)*2;
                                       if(nrb1<a) {fout<<0<<" "; ok=0;}
                                      }
                           }
                }

     if(ok==1)    fout<<1<<" ";
     nrb1=cnrb1;
     nrb2=cnrb2;
     nrb4=cnrb4;
     nrb8=cnrb8;
    }
return 0;
}
```

```
/*
3.5.Autor Adriana Simulescu
*/
#include <fstream>
using namespace std;
ifstream in("taieri.in");
ofstream out("taieri.out");
int n, m;
int main()
{ int ok, i;
  long long n8=0, n4=0,n2=0,n1=0,na=0,nb=0,nc=0,nd=0,a, b, c, d, dif, x;
  in>>n;
  for(i=1; i<=n; ++i)
    {in>>x;
     n8=n8+x/8;
     x=x%8;
     n4=n4+x/4;
```

```
    x=x%4;
    n2=n2+x/2;
    x=x%2;
    n1=n1+x;
   }
  in>>m;
  for(int j=1; j<=m; ++j)
  {
      in>>a>>b>>c>>d;
      nd=n8;nb=nc=na=0;
      if(nd>d)
      {
          dif=nd-d;
          nd=d;
          nc=2*dif;
      }
      nc=nc+n4;
      if(nc>c)
      {
          dif=nc-c;
          nc=c;
          nb=2*dif;
      }
      nb=nb+n2;
      if(nb>b)
      {
          dif=nb-b;
          nb=b;
          na=2*dif;
      }
      na=na+n1;
      ok=1;
      if(nd<d||nb<b||nc<c||na<a) ok=0;
      out<<ok<<" ";
  }
    return 0;
}
```

6.2 Soluții - Clasa a VI-a

1. Butoi

```
/*
1.1. Autor prof. Marinel Șerban
*/

#include <fstream>
#include <cassert>

using namespace std;

ifstream fin("butoi.in");
ofstream fout("butoi.out");

int C, n, k, p, Cv[200005], i, j, sol, cerinta, Smin = 1000000;
int nr_v[10], nr_v_min[10];

int calcul()
{
   int i, S = 0;        //calculez cantitatea de apa
   for (i = 1; i <= n; i++) S += nr_v[i] * Cv[i];
   return S;
}

void adun_1()
{
   //adunare 1 in baza k+1
   int i = n;           //plec de la sfarsit
   while (nr_v[i] == k)//cat timp cifra este k
   {
      nr_v[i] = 0;      //pun 0 in locul ei si
      i--;              //trec mai departe
   }
   nr_v[i]++;           //ultima cifra (pozitia i) o maresc cu 1
}

void retine_min()
{
   int i, S = 0;

   sol++;                //mai am o solutie
   for (i = 1; i <= n; i++) S += nr_v[i];
   if (S < Smin)         //am mai putine operatii?
   {
      Smin = S;          //retin numarul minim de operatii
      for (i = 1; i <= n; i++) nr_v_min[i] = nr_v[i]; //si combinatia
   }
}
```

```
int cerinta3()
{
   int prim = 1, ultim = p, suma = 0, opmin = 1000000, deunde = -1;
   int i, operatii;
   for (i = prim; i <= ultim; i++)
      suma += Cv[i];             //fac suma primelor p galeti
   while (ultim <= n)
   {
      if (C % suma == 0)         //pot umple butoiul complet?
      {
         operatii = C / suma;    //facand de atatea ori operatiile
         if (operatii < opmin)   //sunt mai putine operatii?
         {
            opmin = operatii;    //retine cate si
            deunde = prim;       //de unde incepe secventa
         }
      }
      suma -= Cv[prim];          //scad primul element
      prim++;
      ultim++;                   //treci la secventa urmatoare
      suma += Cv[ultim];         //adaug urmatorul
   }
   return deunde;
}
int main()
{  int rez;
   fin >> cerinta;
   assert(cerinta >= 1 && cerinta <= 3);
   fin >> C >> n >> k >> p;
   assert(5 <= C && C <= 360000);
   if (cerinta < 3)
      assert(1 <= n && n <= 9);
   else
      assert(3 <= n && n <= 100000);
   assert(1 <= k && k <= 5);
   if (cerinta == 3)
      assert(1 <= p && p <= 10000);
   for (i = 1; i <= n; i++)
   {
      fin >> Cv[i];
      if (cerinta < 3)
         assert(1 <= Cv[i] && Cv[i] <= 8000);
      else
         assert(1 <= Cv[i] && Cv[i] <= 200000);
   }
   if (cerinta == 1 || cerinta == 2)
   {
      while (!nr_v[0])           //exista solutie (enunt)
      {
         rez = calcul();         //cantitatea de apa pentru combinatia curenta
         if (rez == C)               //se umple butoiul?
            retine_min();            //retine numarul minim de operatii
         adun_1();                   //trec la alta combinatie de galeti
      }
```

```
      if (cerinta == 1)              //afisari
         fout << sol;                //cerinta 1
      else
         for (i = 1; i <= n; i++)    //cerinta 2
            fout << nr_v_min[i] << ' ';
      fout << '\n';
   }
   else
      fout << cerinta3() << '\n';    //cerinta 3
   return 0;
}
```

```
/*
1.2.Autor prof. Marinel Șerban
solutie fără utilizarea funcțiilor
*/
#include <fstream>
#include <cassert>

using namespace std;

ifstream fin("butoi.in");
ofstream fout("butoi.out");

int C, n, k, p, Cv[200005], i, j, sol, cerinta, rez, S, Smin = 1000000;
int nr_v[10], nr_v_min[10];

int main()
{
   fin >> cerinta;
   //folosesc assert pentru a verifica corectitudinea datelor
   assert(cerinta >= 1 && cerinta <= 3);
   fin >> C >> n >> k >> p;
   assert(5 <= C && C <= 360000);
   if (cerinta < 3)
      assert(1 <= n && n <= 9);
   else
      assert(3 <= n && n <= 100000);
   assert(1 <= k && k <= 5);
   if (cerinta == 3)
      assert(1 <= p && p <= 10000);
   for (i = 1; i <= n; i++)
   {
      fin >> Cv[i];
      if (cerinta < 3)
         assert(1 <= Cv[i] && Cv[i] <= 8000);
      else
         assert(1 <= Cv[i] && Cv[i] <= 200000);
   }
```

```
   if (cerinta == 1 || cerinta == 2)
   {
      while (!nr_v[0])          //exista solutie (enunt)
      {
         rez = 0;            //cantitatea de apa pentru combinatia curenta
         for (i = 1; i <= n; i++)
            rez += nr_v[i] * Cv[i];//calculez cantitatea de apa
         if (rez == C)          //se umple butoiul?
         {
            sol++;              //mai am o solutie
            S = 0;              //numar operatiile
            for (i = 1; i <= n; i++) S += nr_v[i];

            if (S < Smin)       //am mai putine operatii?
            {
               Smin = S;        //retin numarul minim de operatii
               for (i = 1; i <= n; i++)
                  nr_v_min[i] = nr_v[i]; //si combinatia
            }
         }
         //trec la alta combinatie de galeti
         //adunare 1 in baza k+1
         i = n;                 //plec de la sfarsit
         while (nr_v[i] == k)   //cat timp cifra este k
         {
            nr_v[i] = 0;        //pun 0 in locul ei si
            i--;                //trec mai departe
         }
         nr_v[i]++;             //ultima cifra (pozitia i) o maresc cu 1
      }
      //afisez rezultatul la cerintele 1 sau 2
      if (cerinta == 1)
         fout << sol;
      else
         for (i = 1; i <= n; i++)
            fout << nr_v_min[i] << ' ';
      fout << '\n';
   }
   else                         //cerinta 3
   {  int prim = 1, ultim = p, suma = 0, opmin = 1000000, deunde = -1;
      int operatii;
      for (i = prim; i <= ultim; i++)
         suma += Cv[i];         //fac suma primelor p galeti
      while (ultim <= n)
      {  if (C % suma == 0)     //pot umple butoiul complet
         {  operatii = C / suma;//facand de atatea ori operatiile
            if (operatii < opmin)  //sunt mai putine operatii?
            {     opmin = operatii;//retine cate si
                  deunde = prim;   //de unde incepe secventa
            }
         }
         suma -= Cv[prim];      //scad primul element
         prim++;
         ultim++;               //treci la secventa urmatoare
         suma += Cv[ultim];     //adaug urmatorul
      }
      fout << deunde << '\n';
   }
   return 0;
}
```

```
/*
1.3.Autor prof. Marinel Șerban
soluție utilizând sume parțiale
*/
#include <fstream>
#include <cassert>
using namespace std;

ifstream fin("butoi.in");
ofstream fout("butoi.out");
int C, n, k, p, Cv[200005], i, j, sol, cerinta, Smin = 1000000;
int Sp[200005];
int nr_v[10], nr_v_min[10];

int calcul()
{
   int i, S = 0;   //calculez cantitatea de apa
   for (i = 1; i <= n; i++) S += nr_v[i] * Cv[i];
   return S;
}
void adun_1()
{
   //adunare 1 in baza k+1
   int i = n;          //plec de la sfarsit
   while (nr_v[i] == k)//cat timp cifra este k
   {
      nr_v[i] = 0;     //pun 0 in locul ei si
      i--;             //trec mai departe
   }
   nr_v[i]++;          //ultima cifra (pozitia i) o maresc cu 1
}
void retine_min()
{  int i, S = 0;

   sol++;               //mai am o solutie
   for (i = 1; i <= n; i++) S += nr_v[i];
   if (S < Smin)        //am mai putine operatii
   {
      Smin = S;         //retin numarul minim de operatii
      for (i = 1; i <= n; i++) nr_v_min[i] = nr_v[i]; //si combinatia
   }
}
int cerinta3_Sp()
{  int prim = 1, ultim = p, suma = 0, opmin = 1000000, deunde = -1;
   int operatii;
   while (ultim <= n)
   {
      suma = Sp[ultim] - Sp[prim-1];
      if (C % suma == 0)        //pot umple butoiul complet
      {
         operatii = C / suma;   //facand de atatea ori operatiile
         if (operatii < opmin)  //sunt mai putine operatii?
         {
            opmin = operatii;   //retine cate si
            deunde = prim;      //de unde incepe secventa
         }
      }
      prim++;
      ultim++;                  //treci la secventa urmatoare
   }
   return deunde;
}
```

```
int main()
{
   int rez;
   fin >> cerinta;
   assert(cerinta >= 1 && cerinta <= 3);
   fin >> C >> n >> k >> p;
   assert(5 <= C && C <= 360000);
   if (cerinta < 3)
      assert(1 <= n && n <= 9);
   else
      assert(1 <= n && n <= 100000);
   assert(1 <= k && k <= 5);
   if (cerinta == 3)
      assert(1 <= p && p <= 10000);
   for (i = 1; i <= n; i++)
   {
      fin >> Cv[i];
      if (cerinta < 3)
         assert(1 <= Cv[i] && Cv[i] <= 8000);
      else
         assert(1 <= Cv[i] && Cv[i] <= 200000);
      Sp[i] = Sp[i-1] + Cv[i];       //vector cu sume partiale
   }
   if (cerinta == 1 || cerinta == 2)
   {
      while (!nr_v[0])               //exista solutie (enunt)
      {
         rez = calcul();     //cantitatea de apa pentru combinatia curenta
         if (rez == C)               //se umple butoiul?
            retine_min();            //retine numarul minim de operatii
         adun_1();                   //trec la alta combinatie de galeti
      }
      if (cerinta == 1)
         fout << sol;
      else
         for (i = 1; i <= n; i++)
            fout << nr_v_min[i] << ' ';
      fout << '\n';
   }
   else
      fout << cerinta3_Sp() << '\n';
   return 0;
}
```

```
/*
1.4.Autor prof. Marinel Șerban
soluție utilizând numai instrucțiunea for
*/

#include <fstream>
#include <cassert>

using namespace std;

ifstream fin("butoi.in");
ofstream fout("butoi.out");

int C, n, k, p, Cv[200005], i, j, sol, cerinta, Smin = 1000000, rez;
int nr_v[10], nr_v_min[10];
int i1, i2, i3, i4, i5, i6, i7, i8, i9, cate;
```

```
void retine_min1(int S)
{
   sol++;                //mai am o solutie
   if (S < Smin)         //am mai putine operatii
   {
      Smin = S;          //retin numarul minim de operatii
      nr_v_min[1] = i1;
      nr_v_min[2] = i2;
      nr_v_min[3] = i3;
      nr_v_min[4] = i4;
      nr_v_min[5] = i5;
      nr_v_min[6] = i6;
      nr_v_min[7] = i7;
      nr_v_min[8] = i8;
      nr_v_min[9] = i9;
   }
}
void fa_cu_for()
{
   if (n == 9)
   {
      for (i1 = 0; i1 <= k; i1++)
         for (i2 = 0; i2 <= k; i2++)
            for (i3 = 0; i3 <= k; i3++)
               for (i4 = 0; i4 <= k; i4++)
                  for (i5 = 0; i5 <= k; i5++)
                     for (i6 = 0; i6 <= k; i6++)
                        for (i7 = 0; i7 <= k; i7++)
                           for (i8 = 0; i8 <= k; i8++)
                              for (i9 = 0; i9 <= k; i9++)
                              {
                                 rez = i1*Cv[1]+i2*Cv[2]+i3*Cv[3]+i4*Cv[4]
                                          +i5*Cv[5]+i6*Cv[6]+i7*Cv[7]+i8*Cv[8]
                                           +i9*Cv[9];
                                 if (rez == C)          //se umple butoiul?
                                 {
                                    cate = i1+i2+i3+i4+i5+i6+i7+i8+i9;
                                    retine_min1(cate); //retine
                                 }
                              }
   }
   if (n == 8)
   {  for (i1 = 0; i1 <= k; i1++)
         for (i2 = 0; i2 <= k; i2++)
            for (i3 = 0; i3 <= k; i3++)
               for (i4 = 0; i4 <= k; i4++)
                  for (i5 = 0; i5 <= k; i5++)
                     for (i6 = 0; i6 <= k; i6++)
                        for (i7 = 0; i7 <= k; i7++)
                           for (i8 = 0; i8 <= k; i8++)
                           {
                              rez = i1*Cv[1]+i2*Cv[2]+i3*Cv[3]+i4*Cv[4]
                                    +i5*Cv[5]+i6*Cv[6]+i7*Cv[7]+i8*Cv[8];
                              if (rez == C)           //se umple butoiul?
                              {
                                 cate = i1+i2+i3+i4+i5+i6+i7+i8;
                                 retine_min1(cate);   //retine
                              }
                           }
   }
```

```
if (n == 7)
{
   for (i1 = 0; i1 <= k; i1++)
      for (i2 = 0; i2 <= k; i2++)
         for (i3 = 0; i3 <= k; i3++)
            for (i4 = 0; i4 <= k; i4++)
               for (i5 = 0; i5 <= k; i5++)
                  for (i6 = 0; i6 <= k; i6++)
                     for (i7 = 0; i7 <= k; i7++)
                     {
                        rez = i1*Cv[1]+i2*Cv[2]+i3*Cv[3]+i4*Cv[4]
                              +i5*Cv[5]+i6*Cv[6]+i7*Cv[7];
                        if (rez == C)            //se umple butoiul?
                        {
                           cate = i1 + i2 + i3 + i4 + i5 + i6 + i7;
                           retine_min1(cate);    //retine
                        }
                     }
}
if (n == 6)
{
   for (i1 = 0; i1 <= k; i1++)
      for (i2 = 0; i2 <= k; i2++)
         for (i3 = 0; i3 <= k; i3++)
            for (i4 = 0; i4 <= k; i4++)
               for (i5 = 0; i5 <= k; i5++)
                  for (i6 = 0; i6 <= k; i6++)
                  {
                     rez = i1*Cv[1]+i2*Cv[2]+i3*Cv[3]+i4*Cv[4]
                           +i5*Cv[5]+i6*Cv[6];
                     if (rez == C)               //se umple butoiul?
                     {
                        cate = i1 + i2 + i3 + i4 + i5 + i6;
                        retine_min1(cate);       //retine
                     }
                  }
}
if (n == 5)
{
   for (i1 = 0; i1 <= k; i1++)
      for (i2 = 0; i2 <= k; i2++)
         for (i3 = 0; i3 <= k; i3++)
            for (i4 = 0; i4 <= k; i4++)
               for (i5 = 0; i5 <= k; i5++)
               {
                  rez = i1*Cv[1]+i2*Cv[2]+i3*Cv[3]+i4*Cv[4]+i5*Cv[5];
                  if (rez == C)               //se umple butoiul?
                  {
                     cate = i1 + i2 + i3 + i4 + i5;
                     retine_min1(cate);       //retine
                  }
               }
}

if (n == 4)
{
   for (i1 = 0; i1 <= k; i1++)
      for (i2 = 0; i2 <= k; i2++)
         for (i3 = 0; i3 <= k; i3++)
            for (i4 = 0; i4 <= k; i4++)
```

```
                {
                   rez = i1*Cv[1]+i2*Cv[2]+i3*Cv[3]+i4*Cv[4];

                   if (rez == C)                //se umple butoiul?
                   {
                      cate = i1 + i2 + i3 + i4;
                      retine_min1(cate);        //retine
                   }
                }
    }
    if (n == 3)
    {
       for (i1 = 0; i1 <= k; i1++)
          for (i2 = 0; i2 <= k; i2++)
             for (i3 = 0; i3 <= k; i3++)
             {
                rez = i1*Cv[1]+i2*Cv[2]+i3*Cv[3];
                if (rez == C)                //se umple butoiul?
                {
                   cate = i1 + i2 + i3;
                   retine_min1(cate);        //retine
                }
             }
    }
    if (n == 2)
    {
       for (i1 = 0; i1 <= k; i1++)
          for (i2 = 0; i2 <= k; i2++)
          {
             rez = i1*Cv[1]+i2*Cv[2];
             if (rez == C)                //se umple butoiul?
             {
                cate = i1 + i2;
                retine_min1(cate);        //retine numarul minim de operatii
             }
          }
    }
    if (n == 1)
    {
       for (i1 = 0; i1 <= k; i1++)
       {
          rez = i1*Cv[1]+i2*Cv[2];
          if (rez == C)                //se umple butoiul?
          {
             cate = i1;
             retine_min1(cate);        //retine numarul minim de operatii
          }
       }
    }
}
int cerinta3()
{
    int prim = 1, ultim = p, suma = 0, opmin = 1000000, deunde = -1;
    int i, operatii;
    while (ultim <= n)
    {
       suma = 0;
       for (i = prim; i <= ultim; i++)
          suma += Cv[i];        //fac suma celor p galeti
```

```
      if (C % suma == 0)        //pot umple butoiul complet
      {
         operatii = C / suma;   //facand de atatea ori operatiile

         if (operatii < opmin) //sunt mai putine operatii?
         {
            opmin = operatii;  //retine cate si
            deunde = prim;     //de unde incepe secventa
         }
      }
      prim++;
      ultim++;                  //treci la secventa urmatoare
   }
   return deunde;
}

int main()
{
   fin >> cerinta;
   assert(cerinta >= 1 && cerinta <= 3);
   fin >> C >> n >> k >> p;
   assert(5 <= C && C <= 360000);
   if (cerinta < 3)
      assert(1 <= n && n <= 9);
   else
      assert(3 <= n && n <= 100000);
   assert(1 <= k && k <= 5);
   if (cerinta == 3)
      assert(1 <= p && p <= 10000);
   for (i = 1; i <= n; i++)
   {
      fin >> Cv[i];
      if (cerinta < 3)
         assert(1 <= Cv[i] && Cv[i] <= 8000);
      else
         assert(1 <= Cv[i] && Cv[i] <= 200000);
   }
   if (cerinta == 1 || cerinta == 2)
   {
      fa_cu_for();
      if (cerinta == 1)
         fout << sol;
      else
         for (i = 1; i <= n; i++)
            fout << nr_v_min[i] << ' ';
      fout << '\n';
   }
   else
      fout << cerinta3() << '\n';
   return 0;
}
```

```
/*
2.1.Autor Roxana Timplaru
*/
#include <fstream>
#include <iostream>

using namespace std;

ifstream fin("butoi.in");
ofstream fout("butoi.out");

int  gr[100001], k, cerinta,x[10],y[10],n,p;
long long cap;


void citeste()
{ int i;
    fin>>cerinta>>cap>>n>>k>>p;
    for (i=1;i<=n;i++) fin>>gr[i];
}

int mai_pot_aduna()
{
    int i,ok=0;
    for(i=n;i>=1;i--)
        if (x[i]!=k)
              {
                  ok=i;
                  break;
              }
     return ok;
}

int este_capacitate()
{
    int i;
    long long capcalc=0;
    for(i=1;i<=n;i++)
        capcalc+=x[i]*gr[i];
    if (capcalc==cap) return 1;
    else return 0;

}

void norocoasa()
{  int i,j,mini=360000,poz;
   long long s;
    s=0;
    for(i=1;i<=p;i++)
        s=s+gr[i];
    if (cap%s==0)
        {mini=cap/s;
        poz=1;}
    else
        {j=p+1;
    for(i=2;i<=n-p+1;i++)
    {
        s=s-gr[i-1]+gr[j];
        j++;
```

```
        if (cap%s==0 )
             {
                if(cap/s<mini)
                {
                   mini=cap/s;
                  poz=i;
                }
             }
    }
        }
        fout<<poz;
}

void genereaza()
{
    int ok=1,sol=0,i,minim=n*k,s;
    while (ok)
    {
        ok=mai_pot_aduna();
        if (ok)
        {
            x[ok]++;
            for(i=ok+1;i<=n;i++)
                x[i]=0;

        if (este_capacitate())
             {
                 sol++;
                 if (cerinta==2)
                  {
                  for (i=1;i<=n;i++)
                   fout<<x[i]<<" ";
                   break;
                  }
                  s=0;
                  for(i=1;i<=n;i++)
                     s=s+x[i];
                  if (s<=minim)
                  {
                     minim=s;
                     for(i=1;i<=n;i++)
                       y[i]=x[i];
                  }
             }
        }
    }
   if (cerinta==1) fout<<sol;
   }


int main()
{
    citeste();
    if(cerinta!=3)
    genereaza();
    else
        norocoasa();
    return 0;
}
```

2. Păsări

```
/*
2.1.Autor Daniela Lica
*/
#include <fstream>
#include <cassert>

using namespace std;

ifstream f("pasari.in");
ofstream g("pasari.out");

const int Nmax = 1005;

int Cer, N, L, A[Nmax][Nmax];

int V[Nmax], K;
int Nord[Nmax][Nmax], Sud[Nmax][Nmax], Vest[Nmax][Nmax], Est[Nmax][Nmax];

int Mars_L[Nmax][Nmax], Mars_C[Nmax][Nmax];

int main()
{
    f >> Cer >> N;
    assert(Cer == 1 || Cer == 2), assert(5 <= N && N <= 1e3);
    if (Cer == 1)
        f >> L, assert(1 <= L && L <= N);
    for(int i = 1 ; i <= N; ++i)
        for(int j = 1; j <= N; ++j)
            f >> A[i][j], assert(1 <= A[i][j] && A[i][j] <= 1e5);
    if(Cer == 1)
    {
      int c = 0, Max = 0;
      ///Calculam pentru fiecare element de pe linia L nr copaci supravegheati
        for(int C = 1; C <= N; ++C)
        {
            int nr = 1;
            for(int j = C - 1; j >= 1; --j)
                if(A[L][j] <= A[L][C])
                    nr++;
                else
                    break;
            for(int j = C + 1; j <= N; ++j)
                if(A[L][j] <= A[L][C])
                    nr++;
                else
                    break;
            for(int j = L - 1; j >= 1; --j)
                if(A[j][C] <= A[L][C])
                    nr++;
                else
                    break;
```

```
            for(int j = L + 1; j <= N; ++j)
                if(A[j][C] <= A[L][C])
                    nr++;
                else
                    break;
            if(nr > Max)
                Max = nr, c = C;
        }
        g << c << '\n';
        assert(c >= 1 && c <= N);
    }
    else
    {
    ///Pentru fiecare element determinam liniar, in fiecare directie,
    ///în cadrul liniei sau coloanei, pana unde supraveghează
        /// NORD + SUD:
        for(int j = 1; j <= N; ++j)
        {
            /// NORD:
            K = 0;
            for(int i = N; i >= 1; --i)
            {
                while(K && A[i][j] > A[V[K]][j])
                    Nord[V[K]][j] = i, --K;

                V[++K] = i;
            }
            ///
            /// SUD:
            K = 0;

            for(int i = 1; i <= N; ++i)
            {
                while(K && A[i][j] > A[V[K]][j])
                    Sud[V[K]][j] = i, --K;

                V[++K] = i;
            }

            for(int k = 1; k <= K; ++k)
                Sud[V[k]][j] = N + 1;
            ///
        }
        /// VEST + EST:
        for(int i = 1; i <= N; ++i)
        {
            /// VEST:
            K = 0;
            for(int j = N; j >= 1; --j)
            {   while(K && A[i][j] > A[i][V[K]])
                    Vest[i][V[K]] = j, --K;
                V[++K] = j;
            }
```

```
            ///
            /// EST:
            K = 0;
            for(int j = 1; j <= N; ++j)
            {
                while(K && A[i][j] > A[i][V[K]])
                    Est[i][V[K]] = j, --K;

                V[++K] = j;
            }
            for(int k = 1; k <= K; ++k)
                Est[i][V[k]] = N + 1;
            ///
        }
        ///
        /// MARS:
        for(int i = 1; i <= N; ++i)
            for(int j = 1; j <= N; ++j)
            {
                ++Mars_L[i][Vest[i][j] + 1], --Mars_L[i][Est[i][j]];

                ++Mars_C[j][Nord[i][j] + 1], --Mars_C[j][Sud[i][j]];
            }
        for(int i = 1; i <= N; ++i)
            for(int j = 1; j <= N; ++j)
                { Mars_L[i][j] += Mars_L[i][j - 1];
                  Mars_C[i][j] += Mars_C[i][j - 1];
                }
        ///
        int x = 0, y = 0, Max = 0;

        for(int i = 1; i <= N; ++i)
            for(int j = 1; j <= N; ++j)
            {
                int Val = Mars_L[i][j] + Mars_C[j][i] - 1;

                if(Val > Max)
                    Max = Val, x = i, y = j;
            }
        g << x << ' ' << y << '\n';
        assert(1 <= x && x <= N && 1 <= y && y <= N);
    }
    return 0;
}
```

```
/*
2.2.Autor Muntean Radu
*/
#include <iostream>
#include <fstream>
#include <vector>
#include <assert.h>

using namespace std;
```

```
int dezvolt(int n, int i, int j, const vector<vector<int>> &v, int di, int dj)
{
    int ret = 0;
    if (di == 0)
        for (int k = j + dj; k < n && k >= 0; k += dj)
            if (v[i][k] <= v[i][j])
                ret ++;
            else  break;
    if (dj == 0) {
        for (int k = i + di; k < n && k >= 0; k += di)
            if (v[k][j] <= v[i][j])
                 ret ++;
            else break;
    }
    return ret;
}

int cerinta1(int n, int l, const vector<vector<int>> &v) {
    // Indexam linia de la 0.
    l--;
    int max_prj = 0, best_col = 0;
    for (int j = 0; j < n; ++j) {
        int act_prj = 0;
        act_prj += dezvolt(n, l, j, v, 1, 0);
        act_prj += dezvolt(n, l, j, v, -1, 0);
        act_prj += dezvolt(n, l, j, v, 0, 1);
        act_prj += dezvolt(n, l, j, v, 0, -1);
        if (act_prj > max_prj) {
            max_prj = act_prj;
            best_col = j;
        }
    }
    return best_col + 1;
}

void skyline(int n, const vector<vector<int>> &v,
             vector<vector<int>> &lim, int di, int dj) {
    vector<int> stk;
    if (di == 0) {
        int start_j = dj > 0 ? 0 : n-1;
        for (int i = 0; i < n; ++i) {
            for (int j = start_j; j < n && j >= 0; j += dj) {
                while (stk.size() && v[i][stk.back()] < v[i][j]) {
                    stk.pop_back();
                }
                lim[i][j] = stk.size();
                stk.push_back(j);
            }
            while (stk.size()) {
                stk.pop_back();
            }
        }
    } else {
        int start_i = di > 0 ? 0 : n-1;

        for (int j = 0; j < n; ++j) {
            for (int i = start_i; i < n && i >= 0; i += di) {
                while (stk.size() && v[stk.back()][j] < v[i][j]) {
                    stk.pop_back();
                }
```

```
                lim[i][j] = stk.size();
                stk.push_back(i);
            }
            while (stk.size()) {
                stk.pop_back();
            }
        }
    }
}

void cerinta2(int n, const vector<vector<int>> &v, int &best_i, int &best_j) {
    vector<vector<int>> left(n, vector<int>(n));
    vector<vector<int>> right(n, vector<int>(n));
    vector<vector<int>> up(n, vector<int>(n));
    vector<vector<int>> down(n, vector<int>(n));

    skyline(n, v, right,  0, -1);
    skyline(n, v, left, 0,  1);
    skyline(n, v, down,   -1,  0);
    skyline(n, v, up,  1,  0);

    int max_s = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            // cerr << i << " " << j << "\tleft=" << left[i][j] << " \tright="
<< right[i][j] << "\t up=" << up[i][j] << " \tdown=" << down[i][j] << "\n";
            if (right[i][j] + left[i][j] + down[i][j] + up[i][j] > max_s) {
                max_s = right[i][j] + left[i][j] + down[i][j] + up[i][j];
                best_i = i;
                best_j = j;
            }
        }
    }
}

int main() {
    ifstream in ("pasari.in");
    ofstream out ("pasari.out");
    int tip;
    int n, l;

    in >> tip;
    assert(tip == 1 || tip == 2);
    if (tip == 1) {
        in >> n >> l;
        assert(l >= 1);
        assert(l <= n);
    } else {
        in >> n;
    }
    assert(n >= 4);
    assert(n <= 1000);

    vector<vector<int>> v(n, vector<int>(n));

    int val;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            in >> v[i][j];
        }
    }
```

```
    if (tip == 1) {
        out << cerinta1(n, l, v) << std::endl;
    } else {
        int best_i = 0, best_j = 0;
        cerinta2(n, v, best_i, best_j);
        out << best_i + 1 << " " << best_j + 1 << std::endl;
    }

    return 0;
}
```

3. Puternic

```
/*
3.1.Autor Prof. Miana Arisanu
*/
#include <fstream>
#include <cmath>
#define Nmax 4000
using namespace std;

long long  v[100001];
int  prim[600], np;
bool x[Nmax];
ifstream f("puternic.in");
ofstream g("puternic.out");

void Ciur()
{
    int j, i;
    prim[1] = 2;          // in vectorul prim retinem numerele prime < Nmax
    np = 1;                          //numarul numerelor prime < Nmax
    for(i = 3; i*i<=Nmax; i+=2)
        if (x[i]==0)
        {
            for (j=i*i; j<=Nmax; j=j+2*i) x[j]=1;
            prim[++np]=i;
        }
    while (i<=Nmax)
    {
       if (x[i]==0) prim[++np]=i;
       i+=2;
    }
  }
int puternic (long long x)
{
    if (x==1) return 0;
    for (int i=1; i<=np&& x!=1; i++)
    {
        int e=0;
        while (x%prim[i]==0)
        {
            x=x/prim[i];
            e++;
        }
```

```
        if (e==1) return 0;

    }
    if (x==1) return 1;
    long long r2=sqrt(x);                       // radacina patrata a lui x
    if (r2*r2==x ||(r2+1)*(r2+1)==x) return 1;
                  // am evitat eventualele erori de precizie
    long long r3=cbrt(x);                      //radical de ordin 3 din x
    if (r3*r3*r3==x ||(r3+1)*(r3+1)*(r3+1)==x) return 1;
                  // am evitat eventualele erori de precizie
    return 0;
}



long long  concatenare (long long   x, long long  y)
                // concatenarea a doua numere
{
    long long p=1,yy=y;
    while (yy!=0)
    {
        yy=yy/10;
        p=p*10;
    }
    return x*p+y;
}
int main()
{
    int n,p,i,k=0,gasit;
    long long nr,x;
    Ciur();
    f>>p>>n;
    for (i=1; i<=n; i++)
    {
        f>>x;
        if (puternic(x)==0)
        {
            k++; //  numarul valorilor din sir care nu sunt puternice
            v[k]=x;
        }
    }
    if (p==1)
        g<<n-k;
    else
    {
        gasit=0;
        for (i=1; i<=k/2; i++)
        {
            nr=concatenare(v[i],v[k+1-i]);
            if (puternic(nr))
            {
                gasit=1;
                g<<v[i]<<' '<<v[k+1-i]<<'\n';
            }
        }
        if (gasit==0)
            g<<-1;

    }
}
```

```
/*
3.2.Autor Muntean Radu
    O(n * VAL_MAX^(2/5))
*/
#include <fstream>
#include <assert.h>
#include <cmath>
#include <vector>
#include <iostream>
#include <map>

using namespace std;

// sqrt_4(10^18)
const int LIM = 4000;
double eps = 0.00000001;

vector<int> prime;

void ciur() {
    vector<bool> viz(LIM, false);
    for (int i = 2; i < LIM; ++i) {
        if (viz[i]) {
            continue;
        }
        prime.push_back(i);
        for (int k = i * i; k < LIM; k += i) {
            viz[k] = true;
        }
    }
}

bool verif_puternic(long long x) {
    if (x == 1) {
        return false;
    }
    for (int i = 0; i < prime.size(); ++i) {
        if (x % prime[i] == 0) {
            int act_exp = 0;
            while (x % prime[i] == 0) {
                x /= prime[i];
                act_exp ++;
            }
            if (act_exp == 1) {
                return false;
            }
        }
    }
    if (x == 1) {
        return true;
    }
    // Verificam daca x este exact un patrat sau cub perfect.
    long long sq = sqrt(x) + eps;
    if (sq * sq == x) {
        return true;
    }
    long long cb = cbrt(x) + eps;
    if (cb * cb * cb == x) {
        return true;
    }
    return false;
}
```

```
long long lipire_numere(const int &a, const int &b) {
    int aux_b = b;
    long long result = a;
    while (aux_b) {
        aux_b /= 10;
        result *= 10;
    }
    return result += b;
}

int main() {
    ciur();
    ifstream in("puternic.in");
    ofstream out("puternic.out");
    int mode, n;

    in >> mode >> n;
    assert(mode > 0);
    assert(mode < 3);
    assert(n > 0);
    assert(n <= 100000);

    vector<int> v(n);
    for(int i = 0; i < n; ++i)  {
        in >> v[i];
        assert(v[i] > 0);
        assert(v[i] < 1000000000);
    }

    int nr_puternice = 0;
    for (int i = 0; i < n; ++i) {
        if (verif_puternic(v[i])) {
            ++nr_puternice;
            v[i] = -1;
        }
    }

    int last_free = 0;
    for (int i = 0; i < n;  ++i) {
        if (v[i] > 0) {
            v[last_free++] = v[i];
        }
    }
    assert(last_free == n - nr_puternice);

    n = n - nr_puternice;


    if (mode == 1) {
        out << nr_puternice << "\n";
        return 0;
    }

    bool no_answer = true;
```

```
    for (int i = 0; i < n - i - 1; ++i) {
        long long comp = lipire_numere(v[i], v[n - i - 1]);
        if (verif_puternic(comp)) {
            no_answer = false;
            out << v[i] << " " << v[n - i - 1] << "\n";
        }
    }
    if (no_answer) {
        out << "-1\n";
    }

    return 0;
}
```

```
/*
3.3.Autor Daniela Lica
*/

#include <fstream>
#include <cmath>

using namespace std;

ifstream f("puternic.in");
ofstream g("puternic.out");

const int NMAX = 1e5 + 1;
const int VMAX = 31622;

int v[NMAX];

int prim[VMAX + 5], np;
bool x[VMAX + 5];

void Ciur()
{
    for(int i = 3; i * i <= VMAX; i += 2)
        if(!x[i])
            for(int j = i; j * i <= VMAX; j += 2)
                x[i * j] = 1;

    prim[++np] = 2;

    for(int i = 3; i <= VMAX; i += 2)
        if(!x[i])
            prim[++np] = i;

    return;
}

bool puternic_1 (int x)
{
    if(x <= 1)
        return 0;
```

```
    for(int i = 1; i <= np && (prim[i] * prim[i] <= x && x != 1); ++i)
        if(x % prim[i] == 0)
        {
            int e = 0;

            while(x % prim[i] == 0)
                ++e, x /= prim[i];

            if(e == 1)
                return 0;
        }

    return ((x == 1) ? 1 : 0);
}


bool puternic_2 (long long x)
{
    for(int i = 1; i <= np && (prim[i] * prim[i] <= x && x != 1); ++i)
        if(x % prim[i] == 0)
        {
            int e = 0;
            while(x % prim[i] == 0)
                ++e, x /= prim[i];
            if(e == 1)
                return 0;
        }

    if (x == 1)
        return 1;

    long long r2 = sqrt(x);
    if(r2 * r2 == x || (r2 + 1LL) * (r2 + 1LL) == x)
        return 1;

    long long r3 = cbrt(x);
    if(r3 * r3 * r3 == x || (r3 + 1LL) * (r3 + 1LL) * (r3 + 1LL) == x)
        return 1;

    return 0;
}

long long concatenare (long long x, long long y)
{
    long long p = 1, yy = y;

    while(yy)
        yy = yy / 10LL, p = p * 10LL;

    return (x * p + y);
}

int main()
{
    Ciur();

    int p = 0, n = 0, x = 0, k = 0;
```

```
    f >> p >> n;

    for(int i = 1; i <= n; ++i)
    {
        f >> x;

        if(!puternic_1(x))
            v[++k] = x;
    }

    if(p == 1)
        g << n - k << '\n';
    else
    {
        bool gasit = 0;

        for(int i = 1; i <= k / 2; ++i)
        {
            long long nr = concatenare(v[i], v[k - i + 1]);

            if(puternic_2(nr))
            {
                gasit = 1;

                g << v[i] << ' ' << v[k + 1 - i]<< '\n';
            }
        }

        if(gasit == 0)
            g << -1 << '\n';
    }

    return 0;
}
```

```
/*
3.4. Autor Stelian Chichirim
*/
#include <bits/stdc++.h>

using namespace std;

const int Nmax = 1e5, Vmax = 1e9;

vector<int> v;
vector<pair<int, int>> ans;

long long pw(long long x, int p) {
    if (p == 2) return x * x;
    return x * x * x;
}
bool power(long long x) {
    if (x < 2) return false;
    for (int i = 2; 1LL * i * i * i * i * i <= x; ++i) {
```

```
        int nr = 0;
        while (x % i == 0) {x /= i; nr++;}
        if (nr == 1) return false;
    }
    long long c2 = sqrt(x);
    long long c3 = cbrt(x);
    if (pw(c2, 2) == x or pw(c2 + 1, 2) == x or pw(c2 - 1, 2) == x)
        return true;
    if (pw(c3, 3) == x or pw(c3 + 1, 3) == x or pw(c3 - 1, 3) == x)
        return true;
    return false;
}
long long concat(long long x, long long y) {
    long long aux = y;
    while (aux > 0) {
        x *= 10;
        aux /= 10;
    }
    return x + y;
}
int main()
{
    freopen("puternic.in", "r", stdin);
    freopen("puternic.out", "w", stdout);
    int p, n, x;
    scanf("%d", &p);
    assert(1 <= p && p <= 2);
    scanf("%d", &n);
    assert(1 <= n && n <= Nmax);
    int cnt = 0;
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &x);
        assert(1 <= x && x <= Vmax);
        if (power(x)) cnt++;
        else v.push_back(x);
    }
    for (int i = 0; i < v.size() / 2; ++i)
        if (v[i] != v[v.size() - i - 1] && power(concat(v[i],
            v[v.size() - i - 1]))) ans.push_back({v[i], v[v.size() - i - 1]});
    if (p == 1) {
        printf("%d\n", cnt);
        return 0;
    }
    if (ans.size() == 0) printf("-1\n");
    else for (auto r : ans) printf("%d %d\n", r.first, r.second);
    return 0;
}
```

6.3 Soluții - Clasa a VII-a

1. Cat2Pal

```
/*
1.1. Autor prof. Ionel-Vasile Piț-Rada
Complexitate O(log(A)) pentru cerinta 1
Complexitate O(N*sigma*sigma)
*/
#include <fstream>
using namespace std;
ifstream  fin("cat2pal.in");
ofstream fout("cat2pal.out");

int N, v[10002], nw, c, ok, i, j, k, C , P1, P2;
long long int w[100];

long long int z, s, t, s1, s2, t1, t2, x1, x2, x, y, r, X, A, vi,vj, p,q,d;
char f[100002], f0[100002], f1[100002];

int main()
{ fin>>C;
  if (C==1)
     {fin>>A;
     /*voi adauga in w[] toate solutiile posibile, apoi sortez si apoi
     verific daca solutiile posibile sunt printre prefixele sau sufixele
     oglinditului lui A eventual precedate sau succedate de una dintre
     cifrele 0,1,...,9 */
      x=A; while (x%10==0){x=x/10;}; //tai zerourile de la final
      p=1; y=0; do {y=y*10+x%10;x=x/10;p=p*10;} while(x>0); //y=oglindit(A)
      //adaug sufixele si prefixele lui y
      nw=0; q=p; r=y;
      do { w[++nw]=y%q; w[++nw]=r;
           q=q/10; r=r/10;
          } while(r);
      //lipesc cate o cifra la stanga si la dreapta lui y si adaug in w[]
      for (c=0; c<=9; c++) {
          d=y*10+c; if (d<=A*10) {w[++nw]=d;}
          d=c*p+y;  if (d<=A*10) {w[++nw]=d;}
        }
      //sortez candidatii
      for (i=1; i<=nw-1; i++)
          for (j=i+1; j<=nw; j++)
             if (w[i]>w[j]){ int aux=w[i]; w[i]=w[j]; w[j]=aux; }
      //verific si selectez doar solutiile
      w[0]=0; P1=0;
      q=1; while (q<=A) {q=q*10;}
      for (i=1; i<=nw; i++)
        if (w[P1]!=w[i]){ //verific daca w[i] este o solutie
          p=1; while (p<=w[i]){p=p*10;}
          s1=0; x1=A*p+w[i]; t1=x1; do{s1=s1*10+t1%10;t1=t1/10;}while(t1);
          s2=0; x2=w[i]*q+A; t2=x2; do{s2=s2*10+t2%10;t2=t2/10;}while(t2);
```

```
        if (s1==x1 ||s2==x2){ //DA, w[i] este solutie
          w[++P1]=w[i];
        }
      }
    fout<<P1<<"\n";
  }
  if (C==2) {fin>>N; for (i=1;i<=N;i++){ fin>>v[i]; f[v[i]]=1; }
  /* f[x]=1 daca x este din v[]
     orice palindrom de lungime j se poate identifica prin sufix de
     lungime j-j/2
     f0[x]=1 daca x este sufix de palindrom de lungime para
     f1[x]=1 daca x este sufix de palindrom de lungime impara
  */
  //oglinditele prefixelor
  for (i=1; i<=N; i++) {
      vi=v[i]; while (vi && vi%10==0) {vi=vi/10;}
      p=1; y=0; do {y=y*10+vi%10; vi/=10; p=p*10;} while(vi);
      do{
        if (y*10>=p && f[y]==1){//daca y este element din v[]
          z=p*v[i]+y;
          s=z; t=0; j=0; do{j++; t=t*10+s%10;s=s/10;}while(s);
          if (z==t) {//si lipit la dreapta lui v[i] avem palindrom
             //construim in s sufixul identificator pentru z
             k=j-j/2; s=0; while (k) {s=s*10+t%10; t=t/10; k--;}
             if (j%2==0){f0[s]=1;}
                else {f1[s]=1;}
          }
        }
        p=p/10; y=y%p;
      } while(p>=10);
    }
    //oglinditele sufixelor
    for (i=1; i<=N; i++) {
      vi=v[i]; p=1; y=0; do {y=y*10+vi%10;vi=vi/10;p=p*10;} while(vi);
      do {
          if (f[y]==1){  // daca este din v[]
             z=y*p+v[i]; //lipim in stanga lui v[i]
             s=z; t=0; j=0; do {j++; t=t*10+s%10;s=s/10;} while(s);
          if (z==t && f[y]==1){//este palindrom
            //construim in s sufixul identificator pentru z
            k=j-j/2; s=0; while(k) {s=s*10+t%10;t=t/10;k--;}
            if (j%2==0) {f0[s]=1;}
            else {f1[s]=1;}
          }
        }
        y=y/10;
      } while(y>0);
    }
    P2=0;
    for(i=0; i<100000; i++) {P2=P2+f0[i]+f1[i];}
    fout<<P2<<"\n";
  }
  fout.close(); fin.close();
  return 0; }
```

2. Virus

```
/*
2.1. Autor: prof. Adrian Pintea
*/
#include <fstream>
#include <cstring>
using namespace std;
ifstream fin("virus.in");
ofstream fout("virus.out");

int n, rez=0, P;
int cif, poz, maxim,dif;
char cod[201],v[201],cc[201];
int ap[201][53];

int main()
{
    fin>>P>>v>>n;
    for (int j=1; j<=n; j++)
    {
        fin>>cod; poz=0; dif=0;
        if(strlen(cod)==strlen(v))
            for(int k=0; k<strlen(cod); k++)
                if(v[k]!=cod[k])
                {
                    dif++; poz=k;
                    if(cod[k]>='A'&&cod[k]<='Z') cif=26+cod[k]-'A';
                        else cif=cod[k]-'a';
                }
        if (dif==1)
        {
            rez++;
            ap[poz][cif]++;
        }
    }
    if (P==1)
        fout<<rez<<'\n';
    else
    {
        maxim=-1;
        for(int i=0; i<strlen(v); i++)
            for(int j=0; j<52; j++)
                if(ap[i][j])
                    if(ap[i][j]==maxim)
                    {
                        strcpy(cc,v);
```

```
                        if(j<26)cc[i]='a'+j;
                        else cc[i]='A'+j-26;
                        if(strcmp(cc,cod)<0)
                        {
                            maxim=ap[i][j];
                            strcpy(cod,cc);
                        }
                    }
                    else if(ap[i][j]>maxim)
                    {
                        strcpy(cod,v);
                        if(j<26)cod[i]='a'+j;
                        else cod[i]='A'+j-26;
                        maxim=ap[i][j];
                    }
        fout<<cod<<'\n';
    }
    fin.close();
    fout.close();
    return 0;
}
```

3. Zid

```
/*
3.1.Autor prof. Emanuela Cerchez
*/

#include <fstream>
#define NMAX 300
#define CMAX 12

using namespace std;
ifstream fin("zid.in");
ofstream fout("zid.out");
int n, m, c;
int Lmax, imax, jmax;
int Z[NMAX][NMAX];
int F[NMAX][NMAX][CMAX];
int uz[CMAX];

int main()
```

```
{int i, j, k, ii, jj, L, cate;
 //citire
 fin>>n>>m>>c;
 for (i=1; i<=n; i++)
     for (j=1; j<=m; j++) fin>>Z[i][j];
 //determinarea tabloului de frecvente
 // F[i][j][x]=numarul de aparitii ale literei x in zona cu coltul stanga
 // sus (1,1) si coltul dreapta-jos (i,j).
 for (i=1; i<=n; i++)
     for (j=1; j<=m; j++)
         for (k=0; k<=c; k++)
         F[i][j][k]=F[i-1][j][k]+F[i][j-1][k]-F[i-1][j-1][k]+(Z[i][j]==k);

// determinare patrat colorat uniform de arie maxima
 for (i=1; i<=n; i++)
     for (j=1; j<=m; j++)
         for (L=min(n-i+1, m-j+1); L>Lmax; L--)
             {
              ii=i+L-1; jj=j+L-1;
              for (k=0; k<=c; k++)
             uz[k]=F[ii][jj][k]-F[ii][j-1][k]-F[i-1][jj][k]+F[i-1][j-1][k];
              for (k=0; !uz[k]; k++);
              cate=uz[k];
              for (; k<=c; k++)
                  if (uz[k] && uz[k]!=cate) break;
              if (k==c+1) // patrat colorat uniform
                  {if (L>Lmax) {Lmax=L; imax=i; jmax=j;}
                   break;}
             }
 fout<<Lmax*Lmax<<' '<<imax<<' '<<jmax<<'\n';
 fout.close();
 return 0;
}
```

6.4 Soluții - Clasa a VIII-a

1. Bile

```
/**
1.1. Autor Diana Ghinea
      complexitate timp: O(K)
      memorie suplimentară: O(1)
**/
#include <fstream>

using namespace std;

ifstream fin("bile.in");
ofstream fout("bile.out");


int C, N, K;

// numărul X al bilei speciale din cutia A
void c1() {
      if (N % 2 == 1) fout << (N - 1) / 2 << '\n';
      else fout << -1 << '\n';
}

// cel mai mic șir lexicografic
void c2() {
      long long K_patrat = (long long)K * (long long)K;
      long long S = (long long)(K - 2) * (long long)(K - 1) / 2;

      if (K_patrat - S <= N - 1) {
            for (int i = 0; i < K - 1; ++i)
                  fout << i << ' ';
            fout << K_patrat - S << '\n';
            return;
      }

      S += (K - 1);

      int dif = K_patrat - S;
      int g = dif / (N - K);
      int rest = dif % (N - K);

      for (int i = 0; i <= K - (g + 2); ++i)
            fout << i << ' ' ;

      if (K - (g + 1) + rest != K)
            fout << K - (g + 1) + rest << ' ' << N - g << ' ';
      else
            fout << K - (g + 1) + rest + 1 << ' ' << N - g - 1 << ' ';

      for (int i = N - g + 1; i < N; ++i)
            fout << i << ' ';
      fout << '\n';
}
```

```
void c3() {
      if (K % 2 == 0) {
            // K par
            int m = K / 2;
            for (int i = 0; i < K; ++i) {
                  fout << m << ' ';
                  ++m;
                  if (m == K) ++m;
            }
            fout << '\n';
            return;
      }
      // K impar
      int M = K / 2;
      for (int i = 0; i <= M; ++i)
            fout << M + i << ' ';

      for (int i = M - 1; i > 0; --i)
            fout << 2 * K - (M + i) << ' ';
      fout << 2 * K + 1 - M;
      fout << '\n';
}


int main() {
      fin >> C >> N >> K;

      if (C == 1) c1();
      else if (C == 2) c2();
      else if (C == 3) c3();
      return 0;
}

*/
```

```
/**
1.2. Autor Diana Ghinea
    complexitate timp: O(K)
    memorie suplimentară: O(K)
**/
#include <fstream>

using namespace std;

ifstream fin("bile.in");
ofstream fout("bile.out");

const int NMAX = 100000;
const int KMAX = NMAX / 2;

int C, N, K;
int s[KMAX];
void scrie() {
    for (int i = 0; i < K; i++)
        fout << s[i] << ' ';
    fout << '\n';
}
```

```
// numărul X al bilei speciale din cutia A
void c1() {
    if (N % 2 == 1) fout << (N - 1) / 2 << '\n';
    else fout << -1 << '\n';
}

// cel mai mic șir lexicografic
void c2() {
    long long K_patrat = (long long)K * (long long)K;
    long long S = 0;

    for (int i = 0; i < K - 1; i++) {
        s[i] = i;
        S += s[i];
    }

    if (K_patrat - S <= N - 1) {
        s[K - 1] = K_patrat - S;
        scrie();
        return;
    }

    s[K - 1] = K - 1;
    S += s[K - 1];

    int dif = K_patrat - S;
    int g = dif / (N - K);
    int rest = dif % (N - K);

    for (int i = 1; i <= g; i++)
        s[K - i] += (N - K);

    if (s[K - (g + 1)] + rest != K)
        s[K - (g + 1)] += rest;
    else {
        s[K - (g + 1)] += (rest + 1);
        s[K - g] -= 1;
    }

    scrie();
}
void c3() {
    if (K % 2 == 0) {
        // K par
        int m = K / 2;
        for (int i = 0; i < K; ++i) {
            s[i] = m;
            ++m;
            if (m == K) ++m;
        }
        scrie();
        return;
    }
```

```
    // K impar
    int M = K / 2;
    s[0] = M;
    s[K - 1] = 2 * K + 1 - s[0];
    s[M] = K - 1;

    for (int i = 1, j = K - 2; i < M; ++i, --j) {
        s[i] = M + i;
        s[j] = 2 * K - s[i];
    }

    scrie();
}


int main() {
    fin >> C >> N >> K;

    if (C == 1) c1();
    else if (C == 2) c2();
    else if (C == 3) c3();

    return 0;

}
```

```
/*
1.4. Autor Mircea Rotar
*/
#include <bits/stdc++.h>
#define NMAX 100005
using namespace std;
typedef long long ll;
ll a[NMAX]= {};

int main()
{
    ifstream cin("bile.in");
    ofstream cout("bile.out");
    ll C,n,k,med=-1,S,poz=-1,dif,dm,locuri,ok=1,cr=0,minn;
    ll ultim_val;
    cin>>C>>n>>k;

    if(C==1){
        if(n%2!=0)
            med = n/2;
        cout<<med<<endl;
        return 0;
    }
    if(C==2){
    for(int i=0; i<k; i++){
        a[i]=i;
    }
```

```
    dm = n-1-(k-1);
    dif = k*k-k*(k-1)/2;
    locuri = dif/dm;
    for(int i=k-1; i>=(k-1)-locuri+1; i--){
        a[i] = a[i] + dm -(1-ok);
        if(ok==0)
            cr++;
        if(a[i]==k){
            ok=0;
            poz=i;
            a[i]--;
            cr++;
        }
    }
    if(locuri==0) a[k-1]+=dif%dm;
    else{
        ultim_val=a[(k-1)-locuri+1];
        if(dm!=0)
            cr+=dif%dm;

        poz = (k-1)-locuri;

        while(cr>0 && poz>=0){
            minn=min(ultim_val-1-a[poz],cr);
            a[poz]+=minn;
            cr-= minn;
            if(cr!=0)
            poz--;
        }
    }
    if(a[poz]==k){///se verifica daca a[poz]==a[poz+1]
        a[poz]++;
        a[poz+1]--;
    }

    for(int i=0;i<k;i++){
        cout<<a[i]<<" ";
    }
    cout<<endl;
    }
    if(C==3){
        if(k%2==0){
            for(int i=k/2;i<=k-1;i++){
                cout<<i<<" ";
            }
            for(int i=k+1;i<=k+k/2;i++){
                cout<<i<<" ";
            }
            cout<<endl;
        }
        else{
            for(int i=k/2;i<=k-1;i++){
                cout<<i<<" ";
            }
```

```
            for(int i=k+2;i<=k+k/2;i++){
                cout<<i<<" ";
            }
            cout<<k+k/2+2;
            cout<<endl;
        }
    }
    return 0;
}
```

```
/*
1.5. Autor Raluca Costineanu
*/

#include <bits/stdc++.h>
using namespace std;

ifstream f("bile.in");
ofstream g("bile.out");
int C, n, k;

int main()
{
    f>>C>>n>>k;
    if(C==1)
        if(n%2) g<<(n-1)/2<<'\n';
        else g<<-1<<'\n';
    else if(C==2)
    {
        int i,a[50010]= {}, y=n-1;
        long long  x;
        for(i=0; i<=k-1; i++)
            a[i]=i;
        x=1LL*k*k-1LL*(k-1)*k/2;
        for(i=k-1; i>=0 && x>0; i--)
            if(x>=y-i)
            {
                x-=y-i, a[i]=y;
                if(a[i]==k)a[i+1]--, a[i]++;
                y--;
            }
            else
            {
                a[i]+=x;
                if(a[i]==k)a[i+1]--, a[i]++;
                x=0, y--;
            }
        for(i=0; i<k; i++)
            g<<a[i]<<' ';
    }
    else
    {
        int i,a[500010]= {}, p, q, x, y;
        if(k%2==0)
        {
            p=k-1;
            q=k+1;
            x=y=k/2;
```

```
            for(i=x; i>=1; i--)
                a[x--]=p--, a[++y]=q++;
        }
        else
        {
            p=k-1, q=k+2, x=k/2+1, y=k/2+2;
            for(i=x; i>=1; i--)
                a[x--]=p--, a[y++]=q++;
            a[k]++;
        }
        for(i=1; i<=k; i++)
            g<<a[i]<<' ';
    }
    return 0;
}
```

2. Secvente

```
/*
2.1. Autor: Stelian Ciurea
*/
#include <fstream>
#define nmax 1000001
using bigint = long long int;
using namespace std;

bigint  v[nmax], first[nmax], last[nmax], ct[nmax], sumpartiala[nmax],
        sumperest[nmax];
bigint c, n, ctmultipliin, summax;

bool eprima[nmax];

int main()
{
      ifstream in("secvente.in");
      ofstream out("secvente.out");
      in >> c>> n;
      for (int i = 1; i <= n; i++)
            in >> v[i];

      int lungmax = -(n + 1), pozmax, lungcrt;
      fill(first, first + n + 10, -1);
      fill(last, last + n + 10, -1);

      bigint sum = 0, restsum;
      first[0] = 0;
      eprima[0] = 1;
      for (int i = 1; i <= n; i++)
      {
            sum = sum + v[i];
            sumpartiala[i] = sum;
```

```
            restsum = sum % n;
            if (restsum < 0)
                  restsum = restsum + n;
            ct[restsum]++;
            if (first[restsum] < 0)
                  first[restsum] = i;
            last[restsum] = i;
            if (v[i] % n == 0)
                  ctmultipliin++;
            if (restsum != 0)
                  lungcrt = last[restsum] - first[restsum];
            else
                  lungcrt = last[restsum];
            if (lungcrt > lungmax)
            {
                  lungmax = lungcrt;
                  pozmax = restsum;
            }
            bigint sumcrt = sumpartiala[last[restsum]] - sumperest[restsum];
            if (eprima[restsum] == 0)
            {
                  eprima[restsum] = 1;
                  sumperest[restsum] = sumcrt;
            }
            else
            {
                  if (sumcrt < 0)
                        sumperest[restsum] = sumpartiala[last[restsum]];
                  if (sumcrt > summax)
                        summax = sumcrt;
            }
      }

      bigint nrsecvente = 0;
      for (int i = 0; i < n; i++)
            nrsecvente += (ct[i] * (ct[i] - 1) / 2);
      nrsecvente += ct[0] - ctmultipliin;
      if (c == 2)
      {
            out << lungmax << endl;
      }
      if (c == 1)
            out << nrsecvente << endl;
      if (c == 3)
      {     out << summax << endl;
      }
      return 0;
}
```

```cpp
/*
2.2. Autor: Mircea Rotar
*/
#include <bits/stdc++.h>
#define nmax 1000005
using namespace std;
typedef long long ll;
ll maxx, smax=-1e18;
ll a[nmax];
ll rest[nmax];
ll fr[nmax];
ll suma[nmax];
ll sm[nmax];
ll poz[nmax];
unordered_map<ll, ll> um; ///contorizeaza frecventa resturilor

int main()
{
    int C,n;
    ifstream cin("secvente.in");
    ofstream cout("secvente.out");
    cin>>C>>n;
    for(int i=0;i<n;i++){
        cin>>a[i];
    }

    ll sbanal=0,S,SM;  ///sbanal=nr de secvente banale ce se divid cu n

    /// Parcurgem sirul original si calculam:
    /// suma[i] retine suma partiala si anume suma(0..i)
    /// rest[i] retine (suma[0..i] % n)
    /// fr[i] frecventa de aparitie pentru i

    suma[0] = a[0];
    for (ll i = 0; i < n; i++) {
        if(i!=0)
            suma[i] = suma[i-1] + a[i];

        rest[i] = ((suma[i] % n) + n) % n;

        if(a[i]%n==0) {///calculam cate secvente banale avem
            sbanal++;
            smax = max( a[i],smax );
        }
        /// cum suma poate fi negativa se ia modulul de doua ori
        fr[((suma[i] % n) + n) % n]++;
    }
     ll rez = 0;
     for (ll i = 0; i < n; i++){

        if (rest[i] == 0){    /// daca suma(0..i) se divide cu n
            maxx = i + 1;      /// actualizam 'maxx'
            S=suma[i]-suma[poz[rest[i]]];
            SM=sm[rest[i]]+S; ///se foloseste alg lui Kadane
              ///pt a gasi cea mai mare suma a unei secvente

            sm[rest[i]]=max(suma[i],max(SM,S));
            smax=max(smax,sm[rest[i]]);
            poz[rest[i]]=i;
        }
```

```
        /// daca valoarea 'rest[i]' nu este prezenta in 'um'
        /// atunci retinem in 'um' indexul primei aparitii
        else if (um.find(rest[i]) == um.end()){
            um[rest[i]] = i;
            poz[rest[i]]=i;
        }
        else{
            /// in caz de adevar actualizam 'maxx'
            if (maxx < (i - um[rest[i]])){
                maxx = i - um[rest[i]];
            }
            S=suma[i]-suma[poz[rest[i]]];
            SM=sm[rest[i]]+S;
            sm[rest[i]]=max(SM,S);
            smax=max(smax,sm[rest[i]]);
            poz[rest[i]]=i;
        }
        ///daca avem mai multe secvente prefix cu o anumita valoare
        if (fr[i] > 1)
            rez += (fr[i] * (fr[i] - 1)) / 2;
    }
    ///adaugam secventele care sunt divizibile cu n insusi
    ///si cele care au suma = 0
    rez += fr[0];
    rez = rez - sbanal;
    if(C==1)    cout << rez;
    if(C==2)    cout <<maxx;
    if(C==3)    cout<<smax;
    cout<<endl;
    return 0;
}
```

```
/*
2.3 Autor:  Raluca Costineanu
*/
#include <bits/stdc++.h>
using namespace std;
#define Nmax 1000010
ifstream f("secvente.in");
ofstream g("secvente.out");

int C, n, v, poz[Nmax], cate[Nmax], incepe[Nmax], termina[Nmax],
    unde[Nmax], deLa1;
long long s[Nmax], smax[Nmax];
int main()
{
    int i, nrSecv=0, lgMax=0, sumN=0, catiN=0;
    f>>C>>n;
    long long sum=0, sumMax=0;
    for(i=1; i<=n; i++)
    {
        f>>v;   sum+=v; s[i]=sum;
        sumN=sum%n; if(sumN<0)sumN+=n;
```

```
        if(sumN==0){deLa1++; lgMax=i; if(sum>sumMax)sumMax=sum;}

        if(v%n==0)  catiN++;
        cate[sumN]++;
        if(poz[sumN]==0)
            unde[sumN]=poz[sumN]=i;
        else if(i-poz[sumN]>lgMax)lgMax=i-poz[sumN];
        if(cate[sumN]==1)
            incepe[sumN]=termina[sumN]=poz[sumN];
        else
        {
            if(cate[sumN]==2)
                smax[sumN]=sum-s[incepe[sumN]], termina[sumN]=i;
            else
                {
                    if(sum-s[incepe[sumN]]>smax[sumN])
                        smax[sumN]=sum-s[incepe[sumN]],
                        termina[sumN]=i;
                    if(sum-s[unde[sumN]]>smax[sumN])
                        smax[sumN]=sum-s[unde[sumN]],
                        incepe[sumN]=unde[sumN], termina[sumN]=i;
                }
        }
        unde[sumN]=i;
    }
    for(i=0; i<n; i++)
        nrSecv+=cate[i]*(cate[i]-1)/2,
        sumMax=max(sumMax, smax[i]);
    nrSecv=nrSecv-catiN+deLa1;
    if(C==1)g<<nrSecv<<'\n';
    else if(C==2)g<<lgMax<<'\n';
        else g<<sumMax<<'\n';
    return 0;
}
```

```
/*
2.4.Autor Carmen Minca
*/

#include <fstream>

using namespace std;

ifstream f("secvente5.in");
ofstream g("secvente5.out");
long long C,N,x,v[100001], prim[100001],ult[100001], s[100001],
          rest[100001],nrv, nrs,i,sc, r,smax,lgmax,lg,srest[100001],
          sumaseccurenta;
```

```
int main()
{
    f>>C>>N;
    for(i=1;i<=N;i++)
    {
        f>>x;
        if(x%N==0)
            {nrv++;smax=max(smax,x);}
        s[i]=s[i-1]+x;
        r=s[i]%N;
        if(r==0) {smax=max(smax,s[i]);lgmax=max(lgmax,i);}
        if(r<0) r=N+r;
        rest[r]++;
        if (prim[r]==0) ult[r]=prim[r]=i;
        ult[r]=i;
        lg=i-prim[r];
        if(lg>1 && lg>lgmax) lgmax=lg;
        if(rest[r]==1)
             srest[r]=s[i];
        else
          {if(s[i]<srest[r])
               srest[r]=s[i];
           smax=max(smax,s[i]-srest[r]);
             ///suma N-secventei curecte este s[i]-srest[r]
          }
    }

    nrs=rest[0];
    for(i=0;i<N;i++)
        nrs=nrs+rest[i]*(rest[i]-1)/2;
    nrs=nrs-nrv;

    if(C==1)g<<nrs<<endl;
    else
        if(C==2)g<<lgmax<<endl;
    else g<<smax<<endl;
    return 0;
}
```

3. Valoare

```
/*
3.1.Autor Raluca Costineanu
*/
#include <bits/stdc++.h>

using namespace std;

ifstream f("valoare.in");
ofstream g("valoare.out");

int C;
char s[1010];

int main()
{
    f>>C>>s;
    if(C==1)
    {   ///rezolvare cerinta 1
        int i, cate=0;
        bool ap[26]= {};
```

```
        ///marcam in vectorul e aparitii ap literele care apar in sir
        for(i=0; s[i]; i++)
            if(s[i]>='A' && s[i]<='Z')
                ap[s[i]-'A']=1;
        ///determminam cate litere au aprut in sir
        for(i=0; i<26; i++)
            if(ap[i])cate++;
        g<<cate<<'\n';
    }
    else if (C==2)
    {   ///rezolvare cerinta 2
        long long S=0, nr=0;
        int i;
        for(i=0; s[i];)
            if(s[i]>='0' && s[i]<='9')
            {   ///daca in sir apare o cifra
          ///determinam numarul care se formeaza din cifrele care apar alaturat
                while(s[i]>='0' && s[i]<='9')
                    nr=nr*10+(s[i]-'0'), i++;
                S+=nr;
                nr=0;
            }
            else i++;
            g<<S<<'\n';
    }
    else
    {
        ///rezolvare cerinta 3
        long long v[1010]= {-1}, S, nr;
        int i, k=0;
        for(i=0; s[i]; i++)
            if(s[i]=='(')v[++k]=-1; ///adaugam in varful stivei val -1,
                                    ///corespunzatoare inceputului unei repetitii
            else if(s[i]>='A' && s[i]<='Z') ///adaugam in varful stivei
                                           ///valoarea corespunzatoare literei
                    v[++k]=s[i]-'A'+1;
                else if(s[i]==')')///s-a incheiat o secventa care se repeta
                {   ///determinam valoarea subsirului care se repeta
                    S=0;
                    while(k>0 && v[k]!=-1)
                        S+=v[k], k--;
                    v[k]=S;///inlocuim -1, care marca inceputul repetitiei cu
                           ///valoarea subsirului care se repeta
                }
                else if(s[i]>='0' && s[i]<='9')
                {   ///determinam numarul de repetitii a
                    ///subsirului anterior determinat
                    nr=s[i]-'0';
                    while(s[i+1]>='0' && s[i+1]<='9')
                          nr=nr*10+(s[i+1]-'0'), i++;
                    v[k]*=nr;///inlocuim valoarea subsirului care se repeta cu
                            ///valoarea*nr de repetitii
                }
        S=0;    ///calculam valoarea totala a textului
        while(k>0)S+=v[k], k--;
        g<<S<<'\n';
    }
    return 0;
}
```

```
/*
3.2. Autor Mirela Mlisan
*/

#include<cstring>
#include<string>
#include<fstream>
#include<fstream>
using namespace std;
ifstream cin("valoare.in");
ofstream cout("valoare.out");
string s;
int vf,ok, vff;
long long st[1001], stt[1001];
void rezolva_1()
{
    char v[26]= {0};
    int k=0, i;
    for(i=0; i<=s.length(); i++)
        if(s[i]>='A' && s[i]<='Z')
        {
            if(v[s[i]-'A']==0)
            {
                k++;
                v[s[i]-'A']=1;
            }
        }
        else if(s[i]>='0' && s[i]<='9')
            ok=1;
    cout<<k;
}

void rezolva_2()
{
    long long S=0, i=0, x;
    while(i<=s.length())
        if(s[i]>='0' && s[i]<='9')
        {
            x=0;
            while(s[i]>='0' && s[i]<='9'&& i<=s.length())
            {
                x=x*10+s[i]-'0';
                i++;
            }
            S+=x;
        }
        else
            i++;
    cout<<S;
}
void rezolva_3()
{    int S=0, i, n=s.length(), x, ultim=1, j;
    char c;
    for(i=0; i<=n; i++)
        if(s[i]>='0'  && s[i]<='9')
            ok=1;
    if(!ok)
    {
        for(i=0; i<=s.length(); i++)
            if(s[i]>='A' && s[i]<='Z')
                S+=int(s[i]-'A'+1);
```

```
        cout<<S<<'\n';

        return;
    }
  long long sum=0, stot=0;
  i=0;
  while(i<s.length())
    if(isalpha(s[i]))
         {
             sum=0;
              if(vf==0)
                vf=1;
             int l=0,y;
              while(isalpha(s[i]))
              sum+=s[i]-'A'+1, y=s[i]-'A'+1, i++, l++;
              if(isdigit(s[i]))
              {
                  x=0;
                  while(isdigit(s[i]))
                     x=x*10+s[i]-'0',i++;
                     if(l==1)
                  st[vf]=sum*x;
                  else
                     st[vf]=sum-y+y*x;
              }
              else
                 st[vf]=sum;
        vf++;
        st[vf]=0;
         }
  else
     if(s[i]=='(')
     {
         vff++;
         vf++;
         stt[vff]=vf;
         i++;
     }
     else
        if(s[i]==')')
     {
         sum=0;
         for(j=stt[vff];j<=vf;j++)
            sum+=st[j];
        i++;
         if(isdigit(s[i]))
             {
                 x=0;
                 while(isdigit(s[i]))
                    x=x*10+s[i]-'0',i++;
                 vf++;
                 sum=sum*x;
             }
              for(j=stt[vff]+1; j<=vf;j++)
                st[j]=0;
             vf=stt[vff];
             st[vf]=sum;
             vf++;
             vff--;
     }
```

```
for(i=1; i<= vf; i++)
    stot+=st[i];
    cout<<stot;
}

int main()
{
    int ct;
    cin>>ct>>s;
    if(ct==1)
        rezolva_1();
    else if(ct==2)
        rezolva_2();

    else
        rezolva_3();
    return 0;
}
```

```
/*
  3.3. Autor Carmen Minca
*/
#include <fstream>
#include <cstring>
#include <cstdlib>

using namespace std;

ifstream f("valoare.in");
ofstream g("valoare.out");

char s[1005];
long long v[1001];
int n, ps, fr[127];

void suma()
{
    char t[1005]="(", *p, sep[]="ABCDEFGHIJKLMNOPRSTUVXYZWQ()";
    unsigned long long sum=0;
    long long x;

    strcpy(t,s);
    strcat(t,")");
    p=strtok(t, sep);

    while(p)
    {
        x=atoll(p);
        sum+=x;
        p=strtok(NULL,sep);
    }
    g<<sum<<endl;
}

int main()
{
    int c,i;
    long long sc, st=0, nr=0;
    f>>c>>s;
    n=strlen(s);
```

```
    if(c==1)

    {
        for(i=0;i<n;i++)
        {
            fr[s[i]]++;
        }
        int nr=0;
        for(i='A';i<='Z';i++)
            if(fr[i]>0)nr++;
        g<<nr<<endl;
    }
    else
    if(c==2)
        suma();
    else
    if(c==3)
    {
        ps=0;
        i=0;
        while(i<n)
        {
            if (isalpha(s[i]))
            {
                v[++ps]=s[i]-'A'+1;
                i++;
            }
            else if(isdigit(s[i]))
                 {
                    nr=0;
                    while(isdigit(s[i]))
                    {
                       nr=nr*10+s[i]-'0';
                       i++;
                    }
                    v[ps]=v[ps]*nr;
            }
            else if(s[i]=='(')
                 {
                    v[++ps]=-1;
                    i++;
                 }
            else if(s[i]==')')
                 {
                    nr=0;
                    while(v[ps]!=-1)
                    {
                       nr=nr+v[ps];
                       ps--;
                    }
                    v[ps]=nr;
                    i++;
                 }
        }
        st=0;
        for(i=1; i<=ps; i++)
            st=st+v[i];
        g<<st<<endl;
    }
    return 0;
}
```

```
/*
  3.4.Autor Stelian Ciurea
*/

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

string textini, textfinal;
int C;
int lit[200];
long long int rez,suma;
int long long evalstring(string s)
{
    //parsare text fara paranteze
    long long suma = 0, nrcrt=0;
    char litera = s[0];
    for (int i = 1; i < s.size(); i++)
    {
        if (isalpha(s[i]))
        {
            if (nrcrt == 0)
                nrcrt = 1;
            suma += (litera + 1 - 'A')*nrcrt;
            litera = s[i];
            nrcrt = 0;
        }
        if (isdigit(s[i]))
            nrcrt = nrcrt * 10 + (s[i] - 48);
    }
    if (nrcrt == 0)
        nrcrt = 1;
    suma += (litera + 1 - 'A')*nrcrt;
    //cout << suma << endl;
    //system("pause");
    return suma;
}

string evalstringcuparanteze(string s)
{
    int pozparinchisa = s.find(")");
    string aux = s.substr(1, pozparinchisa - 1);
    long long val = evalstring(aux);
    val = val * stoi(s.substr(pozparinchisa + 1));
    //cout << val<<endl;
    string rez = 'A' + to_string(val);
    return rez;
}

int main()
{
    ifstream cin("valoare.in");
    ofstream cout("valoare.out");
    cin >> C;
    cin >> textini;
    if (C == 1)
    {
        for (char c : textini)
            lit[c]++;
```

```
        for (char c = 'A'; c <= 'Z'; c++)
            if (lit[c] > 0)
                rez++;
        cout << rez;
        //system("pause");
        return 0;
    }
    if (C == 2)
    {
        suma = 0;
        string nrcrt;
        for (char c : textini)
        {
            if (isdigit(c))
                nrcrt += c;
            else
            {
                if (nrcrt.size() > 0)
                    suma += stoi(nrcrt);
                nrcrt.clear();
            }
        }
        if (nrcrt.size() > 0)
            suma += stoi(nrcrt);
        cout << suma << endl;
        return 0;
    }
    unsigned poz2, poz1=0, poz3=0;
    do
    {
        poz2 = textini.find(')',poz1);
        if (poz2 >= textini.size())
            break;
        string atom = ")";
        for (int i = poz2-1; i >= 0; i--)
        {

            if (textini[i] == '(')
            {
                atom = '(' + atom;
                poz1 = i;
                break;
            }
            atom = textini[i] + atom;
        }
        for (int i = poz2 + 1; i <= textini.size(); i++)
        {
            if (isdigit(textini[i]))
            {
                atom += textini[i];
                poz3=i;
            }
            else
                break;
        }
        string buf = evalstringcuparanteze(atom);
        textini.erase(poz1, poz3 - poz1 + 1);
        textini.insert(poz1, buf);
    } while (1);
    cout << evalstring(textini) << endl;
}
```

```
/*
  3.5.Autor Stelian Ciurea
*/

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

string  s;
int C,i;
int lit[200];
long long int rez, suma;
char litera;
long long int parsare()
{
    long long nrcrt = 0, suma = 0;
    while (i < s.size())
    {
        if (isdigit(s[i]))
        {
            nrcrt = nrcrt * 10 + (s[i] - 48);
            i++;
        }
        if (isalpha(s[i]))
        {
            if (nrcrt == 0)
                nrcrt = 1;
            suma += (litera + 1 - 'A')*nrcrt;
            litera = s[i];
            nrcrt = 0;
            i++;
        }
        if (s[i] == '(')
        {
            i++;
            if (nrcrt == 0)
                nrcrt = 1;
            suma += (litera + 1 - 'A')*nrcrt;
            litera = 'A' - 1;
            long long rez = parsare();
            suma += rez;
            nrcrt = 0;
        }
        if (s[i] == ')')
        {
            if (nrcrt == 0)
                nrcrt = 1;
            suma += (litera + 1 - 'A')*nrcrt;
            litera = 'A' - 1;
            i++;
            long long nr = 0;
```

```
            while (isdigit(s[i]) and i<s.size())
            {   nr = nr * 10 + (s[i] - 48);
                i++;
            }
            long long temp = suma * nr;
            return temp;
        }
    }
    if (nrcrt == 0)
        nrcrt = 1;
    suma += (litera + 1 - 'A')*nrcrt;
    return suma;
}
int main()
{   ifstream cin("valoare.in");
    ofstream cout("valoare.out");
    cin >> C; cin >> s;
    if (C == 1)
    {
        for (char c : s)  lit[c]++;
        for (char c = 'A'; c <= 'Z'; c++)
            if (lit[c] > 0)   rez++;
        cout << rez;
        //system("pause");
        return 0;
    }
    if (C == 2)
    {
        suma = 0;
        string nrcrt;
        for (char c : s)
        {
            if (isdigit(c))
                nrcrt += c;
            else
            {
                if (nrcrt.size() > 0)
                    suma += stoi(nrcrt);
                nrcrt.clear();
            }
        }
        if (nrcrt.size() > 0) suma += stoi(nrcrt);
        cout << suma << endl;
        //system("pause");
        return 0;
    }
    litera = 'A' - 1;
    s = "(" + s + ")1";
    i = 0;
    cout << parsare() << endl;
    //system("pause");
}
```

6.5. Soluții – Baraj Juniori

1. Intergalactic

```
/* 1.1   Soluție autori */
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;

const int NMAX = 200005;
const int VMAX = 2000005;
bool c[VMAX + 1];
vector <int> prime, comp;
int n, k;

void ciur() {
    c[0] = c[1] = 1;
    for (int i = 4; i <= VMAX; i += 2) {
        c[i] = 1;
    }
    for (int i = 3; i * i <= VMAX; i += 2) {
        if (c[i] == 0) {
            for (int j = i * i; j <= VMAX; j = j + 2 * i) {
                c[j] = 1;
            }
        }
    }
}
bool valid(int sum) {
    int dr = comp.size() - 1, nr = 0;

    for (int st = 0; st < prime.size() && dr >= 0; ++st) {
        while (dr >= 0 && prime[st] + comp[dr] > sum) {
            --dr;
        }
        if (dr < 0) {
            break;
        }
        nr = nr + dr + 1;
        if (prime[st] + comp[dr] == sum) {
            --nr;
        }
        if (nr >= k) {
            return 0;
        }
    }
    return 1;
}
int bs(int st, int dr) {
    int last = st, mij;
    while (st <= dr) {
        mij = (st + dr) / 2;
        if (valid(mij)) {
            last = mij;
            st = mij + 1;
        } else
            dr = mij - 1;
    }
    return last;
}
```

```
int main()
{
    freopen("intergalactic.in", "r", stdin);
    freopen("intergalactic.out", "w", stdout);

    int x, q;
    ciur();

    scanf("%d %d", &n, &q);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &x);
        if (c[x] == 0) {
            prime.push_back(x);
        } else {
            comp.push_back(x);
        }
    }
    sort(prime.begin(), prime.end());
    sort(comp.begin(), comp.end());

    for (int i = 1; i <= q; ++i) {
        scanf("%d", &k);
        printf("%d\n", bs(prime[0] + comp[0], prime[prime.size() - 1] +
              comp[comp.size() - 1]));
    }

    return 0;
}
```

```
/*
 1,2  Soluție autori complexitate O(vmax*log(vmax)+n*log(n)+T*log(vmax)*n)
*/
#include <fstream>
#include <algorithm>
using namespace std;
ifstream  fin("intergalactic.in");
ofstream fout("intergalactic.out");
int T, n, k, a[200002], b[200002], aux, p, q, i, j, *x, *y, z;
char ciur[2000002];
long long f(int s){
    ///calculeaza cate perechi au suma <=s
    long long  nr=0;
    int i,j1,j2,js=q;
    for(i=1;i<=p && js>0;i++){
        if(i>=2 && x[i]-x[i-1]<100000)
            while(js>0 && x[i]+y[js]>s)js--;
        else{
            j1=1; j2=js; js=0;
            while(j1<=j2){
                j=(j1+j2)/2;
                if(x[i]+y[j]<=s){
                    js=j;
                    j1=j+1;
                }
                else j2=j-1;
            }
        }
        nr=nr+js;
    }
    return nr;
}
```

```
int main(){
    ciur[1]=1;
    for(i=2;i*i<=2000000;i++){
        if(ciur[i]==0)
            for(j=i*i;j<=2000000;j=j+i)
                ciur[j]=1;
    }
    fin>>n>>T;
    p=0;q=0;
    for(i=1;i<=n;i++){
        fin>>z;
        if(ciur[z]==0){
            p++;a[p]=z;
        }
        else{
            q++;b[q]=z;
        }
    }
    sort(a+1,a+1+p);
    sort(b+1,b+1+q);
    x=a; y=b;
    if(p>q){x=b;y=a;aux=p;p=q;q=aux;}
    for(int t=1;t<=T;t++){
        fin>>k;
        long long k1;
        int s1,s2,s3=0,s;
        s1=x[1]+y[1]; s2=x[p]+y[q];
        while(s1<=s2){
            s=(s1+s2)/2;
            k1=f(s);
            if(k1>=k){
                s3=s;
                s2=s-1;
            }
            else
                s1=s+1;
        }
        fout<<s3<<"\n";
    }

    return 0;
}
```

```
/* 1.3.    Soluție autori */

#include <bits/stdc++.h>
#define nmax 2000002
using namespace std;

ifstream fin("intergalactic.in");
ofstream fout("intergalactic.out");

bitset<nmax> b;
int n;
int B[nmax], nb; /// se retin numerele neprime
int A[nmax], na; /// se retin numerele prime
```

```
void Eratostene(int n)
{
    int i, j;
    b[0] = b[1] = 1;
    for (i = 4; i <= n; i += 2)
        b[i] = 1;
    for (i = 3; i * i <= n; i += 2)
        if (b[i] == 0)
            for (j = i * i; j <= n; j += 2 * i)
                b[j] = 1;
}
long long NumaraSume(int suma)
{
    if (A[1] + B[1] > suma) return 0;
    if (A[na] + B[nb] <= suma) return 1LL * na * nb;
    int i, j;
    long long cnt = 0;
    j = na;
    for (i = 1; i <= nb && j >= 1; i++)
    {
        while (j >= 1 && A[j] + B[i] > suma)
            j--;
        cnt += j;
    }
    return cnt;
}
int main()
{
    int i, x, T, k, st, dr, sol, suma;
    long long cnt;
    Eratostene(2000000);
    fin >> n >> T;
    for (i = 1; i <= n; i++)
    {
        fin >> x;
        if (b[x] == 0) A[++na] = x;
        else B[++nb] = x;
    }
    /// sortam numerele prime si cele neprime
    sort(A + 1, A + na + 1);
    sort(B + 1, B + nb + 1);
    while (T--)
    {
        fin >> k;
        /// cautare binara a celei de k-a sume
        st = 0; dr = 1e9; sol = dr;
        while (st <= dr)
        {
            suma = (st + dr) / 2;
            cnt = NumaraSume(suma);
            if (cnt >= k)
            {
                sol = suma;  dr = suma - 1;
            }
            else st = suma + 1;
        }
        fout << sol << "\n";
    }
    fin.close(); fout.close();
    return 0;
}
```

```
/*
 1.4. Autor: Raluca Costineanu
*/
//Raluca Costineanu - ciur pt verificarea primalitatii
//+ Cautare binara pentru gasirea valorii
#include <bits/stdc++.h>
using namespace std;
#define nMax 200010
ifstream f("intergalactic.in");
ofstream g("intergalactic.out");
int N, T, prime[nMax], compuse[nMax], n, m ;
bool prim[2000010];

void ciur()
{
    int i, j;
    prim[0]=prim[1]=1;
    for(i=3; i<=1415; i+=2)
        if(prim[i]==0)
        for(j=i*i; j<=2000000; j+=i)
        prim[j]=1;
}
bool bun(int val, int k)
{   int i=1, j=m, cate=0;
    while(i<=n)
    {
        if(prime[i]+compuse[j]<val)
            cate+=j;
        else
        {
            cate+=j;
            while(j>=1 && prime[i]+compuse[j]>=val)
                j--, cate--;
        }
        if(cate>=k)return 0;
        i++;
    }
    return 1;
}
int valoarea(int k)
{   int st=prime[1]+compuse[1], dr=prime[n]+compuse[m], mij, val=0;
    while(st<=dr)
    {   mij=(st+dr)/2;
        if(bun(mij, k))
            val=mij, st=mij+1;
        else dr=mij-1;
    }
    return val;
}
```

```
int main()
{
    f>>N>>T;
    int x, i;
    ciur();
    for(i=1; i<=N; i++)
    {
        f>>x;
        if(x==2)prime[++n]=x;
        else if(x%2==0)compuse[++m]=x;
        else if(prim[x]==0)prime[++n]=x;
        else compuse[++m]=x;
    }
    sort(prime+1, prime+n+1);
    sort(compuse+1, compuse+m+1);
    for(i=1; i<=T; i++)
        f>>x, g<<valoarea(x)<<'\n';
    return 0;
}
```

2. Cârtița

```
/*
  2.1. Autor: Dana Lica
*/
///O(U + N*log2N + Q)
#include <fstream>
#include <cassert>
using namespace std;

ifstream f("cartita.in");
ofstream g("cartita.out");

const int NMAX = 1e5 + 1, LOGMAX = 17;
const long long INF = 1LL * 1e18;
int N, U, Q;
long long A[NMAX];
long long Sum_X = 0, Sum_K = 0;
long long Diff = 0;
int Log2[NMAX];
long long rmq[LOGMAX][NMAX];

void Read ()
{
    f >> N;
    assert(1 <= N && N <= 1e5);
    for(int i = 1; i <= N; ++i)
        f >> A[i], assert(1 <= A[i] && A[i] <= N);
    return;
}
```

```
void Update ()
{
    int i = 0, x = 0, K = 0;
    f >> i >> x >> K;
    assert(1 <= i && i <= N);
    assert(1 <= x && x <= 400);
    assert(-21 <= K && K <= 21 && K);
///    assert(1 <= x && x <= 1e3);
///    assert(1 <= K && K <= 21);
    Sum_X += 1LL * x, Sum_K += 1LL * K;
    Diff += 1LL * i * K;
    return;
}

long long my_min (long long a, long long b)
{
    return ((a < b) ? a : b);
}

void Initialize ()
{
    Log2[1] = 0;
    for(int i = 1; i <= N; ++i)
    {
        if(i > 1)
            Log2[i] = 1 + Log2[(i >> 1)];
        rmq[0][i] = A[i];
    }
    for(int i = 1; i <= Log2[N]; ++i)
    {
        int Lg = (1 << i);
        for(int j = 1; j <= (N - Lg + 1); ++j)
            rmq[i][j] = my_min(rmq[i - 1][j], rmq[i - 1][j + (Lg >> 1)]);
    }
    return;
}
void Updates ()
{
    f >> U;
    assert(1 <= U && U <= 3e5);
    while(U--)
        Update();
    for(int i = 1; i <= N; ++i)
        A[i] += 1LL * Sum_X + 1LL * i * Sum_K - Diff;
    Initialize();
    return;
}
long long Query_Min (int left, int right)
{
    int k = Log2[(right - left + 1)];
    return my_min(rmq[k][left], rmq[k][right - (1 << k) + 1]);
}
```

```
void Query ()
{   int left = 0, right = 0;
    f >> left >> right;
    assert(1 <= left && left <= right && right <= N);
    g << Query_Min(left, right) << '\n';
    return;
}
void Queries ()
{   f >> Q;
    assert(1 <= Q && Q <= 2e5);
    while(Q--)
        Query();
    return;
}
int main()
{   Read();
    Updates();
    Queries();
    return 0;
}
```

```
/*
2.2. Autor: Ioan Cristian Pop
*/

#include <cstdio>
using namespace std;
const int NMAX = 100000;
const int LGMAX = 20;

long long rmq[LGMAX + 2][NMAX + 5];
long long v[NMAX + 5];
int lg[NMAX + 5], n;

long long minim (long long a, long long b) {
    if (a < b) {
        return a;
    }
    return b;
}

void solve_rmq() {
    for (int i = 1, k = 0; i <= n; ++i) {
        if (i >= (2 << k)) {
            ++k;
        }
        lg[i] = k;
    }

    for (int i = 1; i <= n; ++i) {
        rmq[0][i] = v[i];
    }
```

```
    for (int i = 1; i <= lg[n]; ++i) {
        for (int j = 1; j <= n; ++j) {
            if ((1 << i) <= j) {
              rmq[i][j] = minim(rmq[i - 1][j], rmq[i - 1][j - (1 << (i - 1))]);
            }
        }
    }
}

int main()
{
    freopen("cartita.in", "r", stdin);
    freopen("cartita.out", "w", stdout);

    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        scanf("%lld", &v[i]);
    }

    long long sx = 0, sk = 0, diff = 0;
    int poz = 0, x, k, u;

    scanf("%d", &u);
    for (int i = 1; i <= u; ++i) {
        scanf("%d %d %d", &poz, &x, &k);
        sx += 1LL * x;
        sk += 1LL * k;
        diff += 1LL * poz * k;
    }

    for (int i = 1; i <= n; ++i) {
        v[i] += sx + 1LL * i * sk - diff;
    }
    solve_rmq();

    int st, dr, q, mid;

    scanf("%d", &q);
    for (int i = 1; i <= q; ++i) {
        scanf("%d %d", &st, &dr);
        mid = lg[dr - st + 1];
        printf("%lld\n", minim(rmq[mid][dr], rmq[mid][st + (1 << mid) - 1]));
    }
    return 0;
}
```

```
/*
2.3. Autor: Victor Manz - Colegiul Național de Informatică „Tudor Vianu”
*/

#include <fstream>
#include <iostream>
using namespace std;

const int N = 1e5;
const int L = 16;
int v[N+1], n, u, q, r[L+1][N+1], log2[N+1];

int raspuns(int st, int dr)//min(v[st], v[st+1], ..., v[dr])
{
    //2^e = cea mai mare putere a lui 2 mai mica sau egala cu dr - st + 1
    int e = log2[dr - st + 1];
    int p2 = 1 << e;
    return min(r[e][st + p2 - 1], r[e][dr]);
}

int main()
{
    ifstream in("cartita.in");
    ofstream out("cartita.out");
    in >> n;
    for (int i = 1; i <= n; i++)
    {
        in >> v[i];
    }
    in >> u;
    int sum = 0, sk = 0;
    for (int i = 0; i < u; i++)
    {
        int pos, x, k;
        in >> pos >> x >> k;
        sum += x - (pos - 1) * k;
           //suma expresiilor de forma (pos - 1) * k se va aduna
           //la fiecare element al sirului
        sk += k;
        //suma k-urilor se va aduna la al i-lea element inmultita cu (i-1)
    }
    //actualizam v: obtinem starea acestuia dupa efectuarea celor u operatii
    //initializam linia 0 a matricei r (structura de date de tip RMQ)
    //r[i][j] =minimul subsecventei de lungime 2^i care se termina pe pozitia j
```

```
    for (int i = 1; i <= n; i++)
    {
        v[i] += sum + sk * (i - 1);//actualizarea
        r[0][i] = v[i];//initializare necesara pentru RMQ
            //r[0][j] = minimul subsecventei de lungime 1 care
            //se termina pe pozitia j
    }
    //precalculam vectorul log2, cu log2[i] = parte intreaga din
    //logaritm in baza 2 din i
    log2[1] = 0;
    for (int i = 2; i <= n; i++)
    {
        log2[i] = 1 + log2[i/2];
    }
    //completam urmatoarele linii ale matricei r
    for (int i = 1; i <= L; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            if ((1 << i) <= j)
            {
                r[i][j] = min(r[i-1][j - (1 << (i-1))], r[i-1][j]);
            }
        }
    }
    //raspundem la intrebari
    in >> q;
    for (int i = 0; i < q; i++)
    {
        int st, dr;
        in >> st >> dr;
        out << raspuns(st, dr) << "\n";
    }
    in.close();
    out.close();
    return 0;
}
```

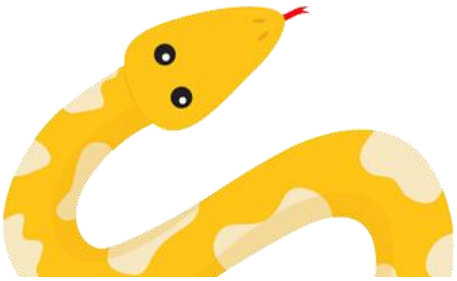
3.Inno

```
/*
   Autor: Dan Pracsiu
*/

#include <bits/stdc++.h>
#define inFile "inno.in"
#define outFile "inno.out"
#define nmax 200003
using namespace std;

int a[nmax], st[nmax], dr[nmax], n, k;
long long nrSecv;

int NrBiti_1(int n)

{
    int s = 0;
    while (n > 0)
    {   n = (n & (n - 1));
        s++;
    }
    return s;
}

void Citire()
{
    ifstream fin(inFile);
    fin >> n >> k;
    for (int i = 1; i <= n; i++)
        fin >> a[i];
}

void Rezolvare()
{
    int i, j, x, y, stop;
    ofstream fout(outFile);
    x = 0;
    st[0] = (1 << 30) - 1;
    stop = 0;
    for (i = 1; i <= n && !stop; i++)
    {   st[i] = st[i - 1] & a[i];
        if (NrBiti_1(st[i]) >= k) x = i;
        else stop = 1;
    }
    if (x == n)
    {   nrSecv = 1LL * n * (n + 1) / 2 - 1;
        fout << nrSecv << "\n";
        fout.close();
        return;
    }
    y = n + 1;
```

```
    dr[n + 1] = (1 << 30) - 1;
    stop = 0;
    for (i = n; i >= 1; i--)
    {   dr[i] = dr[i + 1] & a[i];
        if (NrBiti_1(dr[i]) >= k) y = i;
        else stop = 1;
    }
    if (x == 0 && y == n + 1) /// nu sunt solutii
    {   fout << "0\n";
        fout.close();
        return;
    }
    if (y == n + 1)
    {   fout << x << "\n";
        fout.close();
        return;
    }
    if (x == 0)
    {   fout << (n - y + 1) << "\n";
        fout.close();
        return;
    }
    j = y;
    nrSecv = (n - y + 1);
    for (i = 1; i <= x && j <= n; i++)
    {   while (j <= n && NrBiti_1(st[i] & dr[j]) < k)
            j++;
        nrSecv += (n - j + 2);
    }
    fout << nrSecv << "\n";
    fout.close();
}

int main()
{
    Citire();
    Rezolvare();
    return 0;
}
```

Editura L&S Soft | Infobits Academy
https://www.infobits.ro

Cărți în format electronic cu specific informatic
https://ebooks.infobits.ro



Curs online gratuit interactiv – limbajul Python 3
https://www.pythonisti.ro