Descrierea generală a problemelor

Nr. crt.	Denumirea problemei	Restricția referitoare la volumul utilizat de memorie	Restricția referitoare la timpul de execuție, secunde	Punctajul alocat problemei
1.	Spargerea codului	≤ 4Mb	≤ 0,2	100
2.	Robotul	≤ 1Mb	≤ 0,1	100
3.	Parole	≤ 6Mb	≤ 0,1	100

Notă. În caz de egalitate de punctaj, prioritate se va da concurentului care a obținut punctajul respectiv primul.

Spargerea codului

Factorizarea unui număr **N** înseamnă reprezentarea sa ca produs de numere prime. De exemplu, numărul 15 poate fi reprezentat ca produsul dintre 3 și 5. Este foarte simplu să verificăm că 3 și 5 sunt numere prime și că produsul lor este 15. Cu toate acestea, operația inversă, adică factorizarea unui număr, este o sarcină dificilă. Sistemele criptografice moderne se bazează pe dificultatea factorizării numerelor mari.

Sarcină. Scrieți un program care să poată sparge astfel de sisteme. Vi se dă un număr **N**, care este produsul a două numere prime. Găsiți aceste numere. Dacă aceste numere nu există, afișați `failed`.

Date de intrare. Prima linie a intrării standard va conține un număr natural **N**.

Date de ieșire. În prima linie a ieșirii standard afișați două numere prime separate printr-un spațiu, ordonate în ordine crescătoare, care atunci când sunt înmulțite dau **N**. Dacă aceste numere nu există, afișați `failed`.

Restricții. $1 <= N <= 10^{12}$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `spargere.pas`, `spargere.c` sau `spargere.cpp`.

Exemplu

Intrare	Ieșire
15	3 5

Intrare	Ieșire
10	2 5

Intrare	Ieșire
7	failed

Robotul

Un grup de cercetători studiază planetele îndepărtate, iar pentru a culege de pe acele planete careva probe de sol, ei sunt nevoiți să utilizeze un robot controlat de la distanță. Înainte de a folosi robotul, cercetătorii capturează cu ajutorul satelitului o reprezentare a planetei ce va fi studiată. Reprezentarea suprafeței planetei este descrisă cu ajutorul unui tablou bidimensional de *n* rânduri și *m* coloane format din zerouri (0) și unități (1). Unde *0* ar semnifica zone pe care robotul se poate deplasa, iar zonele marcate cu *1* reprezintă zonele pe care robotul nu se poate deplasa.

Robotul este nevoit să traverseze planeta din colțul stâng-sus până în colțul drept-jos utilizând cea mai scurtă cale posibilă. Se știe că robotul se poate deplasa pe direcțiile: Nord și Sud (vertical), Est și Vest (Orizontal), Nord-Vest, Nord-Est, Sud-Vest și Sud-Est (Pe diagonale). Se știe că deplasarea de pe o coordonată pe alta este considerată drept un pas efectuat. Aterizarea pe coordonata de start la fel este considerată ca un pas efectuat.

Sarcină. Elaborați un program care determină numărul minim de pași necesari de efectuat din colțul stâng-sus pînă în colțul drept-jos într-un tablou bidimensional, urmând regulile descrise mai sus.

Date de intrare. Intrarea standard pe primul rând conține două numere întregi *n* *și* *m* separate prin spațiu, unde *n* reprezintă numărul de rânduri, iar *m* numărul de coloane. Apoi pe *n* rânduri urmează *m* numere întregi (0 sau 1) ce formează harta planetei date.

Date de ieșire. Ieșirea standard va conține un număr întreg care reprezintă numărul minim de pași necesari pentru a traversa planeta. Dacă este imposibil de construit traseul, programul va afișa `-1`.

Restricții. `1≤n≤100, 1≤m≤100.` Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea *robotul*`.pas, `*robotul*`.c sau `*robotul*`.cpp.`

Exemplu.

Intrare

```
4 4
0 1 0 1
1 0 0 1
1 0 1 1
0 0 0 0
```

Ieșire

```
5
```

Explicație. Pentru a ajunge din colțul stâng-sus a tabloului în colțul drept-jos a tabloului este necesar de 5 pași. Primul pas fiind aterizarea, al doilea pas fiind deplasarea Sud-Est (pe diagonală), al treilea pas deplasarea în regiunea Sud, al patrulea pas deplasarea Sud-Est (pe diagonală) și ultimul pas deplasarea pe Est.

Intrare

```
2 3
1 0 1
0 1 1
```

Ieșire

```
-1
```

Explicație. Este imposibil de ajuns la punctul final.

Parole

La o întreprindere secretă generarea parolelor este un lucru foarte important care trece printr-o serie de verificări și testări. În acest sens există o politică specială de validare a parolelor în cadrul companiei date. Parola poate fi compusă doar din literele: `a, e, i, o, u` (toate minuscule). Și există următoarele reguli după care poate fi formată o parolă.

- După litera 'a' poate urma doar litera 'e';
- După litera 'e' poate urma doar litera 'a' sau 'i';
- După litera 'i' nu poate urma o altă literă 'i';
- După litera 'o' poate urma doar litera 'i' sau 'u';
- După litera 'u' poate urma doar litera 'a'

Astfel, se cere de aflat câte parole unice de lungimea N pot fi construite urmând regulile descrise.

Sarcină. Elaborați un program care determină numărul de parole unice ce pot fi construite urmând regulile descrise. Deoarece numărul poate fi extrem de mare, prezentați-l redus după aplicarea operației $\bmod 10^9 + 7$.

Date de intrare. Intrarea standard va conține un număr întreg N ce reprezintă numărul de litere prezent în parolă.

Date de ieșire. Ieșirea standard va conține un număr întreg (redus prin $\bmod 10^9 + 7$) care reprezintă câte parole unice pot fi generate.

Restricții. $1 \leq N \leq 10^4$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea *parole*.pas, *parole*.c sau *parole*.cpp.

Exemplu.

Intrare	Ieșire
2	10

Explicație. Pot fi generate 10 parole de caractere unice. Ele sunt: ae, ea, ei, ia, ie, io, iu, oi, ou, ua.