COVID tests

La școala lui Adam, a apărut un nou val pandemic COVID. Pentru a preveni răspândirea pandemiei, școala a decis să testeze toți elevii folosind testul antigen bazat pe saliva acestora.

Cum toți profesorii au uitat cum se fac aceste teste, Adam s-a oferit să îi ajute.

El a primit teste de saliva de la cei N elevi (din motive de securitate, are voie să cunoască doar codul acestora, cuprins între 0 și $N-1$), iar sarcina sa este să determine care teste sunt pozitive. Din păcate, a fost prea târziu când Adam a realizat că testarea elevilor este o sarcină dificilă și plictisitoare. Cu toate acestea, el a realizat că poate face testarea într-un mod mai inteligent decât testând secvențial. Dacă amestecă o submulțime de probe și testează această combinație, el va ști dacă toate probele sunt negative sau dacă cel puțin una dintre ele este pozitivă. El se poate folosi de această metodă pentru a reduce numărul de testări.

Cum este suficientă salivă în fiecare probă, el poate folosi o probă de oricâte ori. Cu atât mai mult, testele sunt absolut precise, deci nu se poate întâmplă niciodată să se obțină răspunsuri diferite pentru aceeași probă.

În aceste condiții, el dorește să optimizeze acest proces prin a folosi cât mai puține teste cu putință. Fiind ocupat cu testarea, vă lasă pe voi să vă ocupați de această optimizare.

Conform statisticilor locale, el știe că probabilitatea ca o probă să fie pozitivă în urma testului este egală cu P, iar această probabilitate nu este influențat în niciun fel de faptul că alte probe sunt pozitivă sau negative. Se poate să folosiți această informație pentru a optimiza ce teste va face Adam?

Comunicare

Problema este interactivă.

Programul tău va fi executat pe mai multe teste.

La fiecare rulare a programului, se vor rezolva T scenarii diferite, toate acestea constituind un singur test.

Valorile lui N și a lui P sunt aceleași pentru fiecare dintre scenarii, dar probele care ies pozitive pot fi diferite de la un scenariu la altul.

Se poate implementa protocolul descris mai jos, sau se poate folosi cel deja implementat. Poți găsi implementarea în CMS ca un atașament numit `template.cpp`.

Protocolul de comunicare

La început, programul ar trebui să citească de pe prima linie din standard input un număr întreg N, un număr real P și un număr întreg T separate prin câte un spațiu — numărul de elevi, probabilitatea ca o probă să fie pozitivă și numărul de scenarii.

După aceea, programul poate afișa interogări la standard output. Fiecare interogare constă dintr-o singură linie care conține `Q`, un spațiu, și un șir s de lungime N, unde s_i este `1` dacă proba pentru a i-a persoană face parte din test, sau `0` altfel. După afișarea acestei linii, programul va trebui să dea flush la standard output și apoi să citească un singur caracter de pe o singură linie care va fi `P` dacă cel puțin o probă din grupul testat este pozitivă, sau `N` altfel.

Programul poate să afișeze la standard output răspunsul pe o singură linie care va conține `A`, un spațiu și un șir de caractere s de lungime N, unde s_i este `1` dacă proba celei de-a i-a persoană este pozitivă, sau `0` altfel. După afișarea aceste linii, programul trebuie să dea flush la standard output, iar apoi să citească un caracter de pe o singură linie.

Dacă caracterul de pe linia citită este `C`, atunci răspunsul este corect. În acest caz programul poate să pună interogări pentru scenariul următor, sau, dacă acesta a fost al T-lea răspuns, programul trebuie oprit.

Dacă caracterul de pe linia citită este `W`, atunci răspunsul nu a fost corect. În acest caz, programul trebuie terminat imediat.

Este important ca programul să se oprească după citirea caracterului `W` pentru a primi un feedback corect din partea CMS. Dacă programul tău continuă să ruleze, atunci s-ar putea să dea crash sau să primească alte verdicte greșite.

Implementarea protocolului

Dacă folosiți implementarea protocolului din `template.cpp`, trebuie să implementați funcția `std::vector<bool> find_positive()`. Această funcție va fi apelată o dată pentru fiecare scenariu. Funcția trebuie să returneze un vector de N variabile boolene, unde al i-lea element este `true` dacă și numai dacă proba celui de-al i-lea elev este pozitivă.

Se poate folosi funcția `bool test_students(std::vector<bool> mask)` pentru a determina cine este pozitiv. Această funcție execută un test pe un amestec de probe. Singurul argument al funcției este un vector de N variabile boolene, unde al i-lea element este adevărat dacă a i-a probă trebuie adăugată în amestec. Funcția întoarce valoarea `true` dacă și numai dacă cel puțin una dintre probe din amestec este pozitivă.

Puteți folosi variabilele globale `N` și `P` care conțin valorile lui N și P din enunț. Puteți face orice inițializare în funcția `main` după primul apel al funcției `scanf`.

Inputs

Evaluarea pentru această problemă nu este adaptivă, ceea ce înseamnă că probele pozitive sunt determinate înainte de începerea rulării programului. Cu atât mai mult, pentru fiecare probă s-a determinat dacă este pozitivă în mod independent cu o probabilitate P folosind un generator aleator corect de numere.

Grupe de teste și evaluare

Sunt două grupe de teste.

Prima grupă (10 points)

- $N = 1\,000$
- $T = 1$
- $0 \leq P \leq 1$

O soluție este acceptată dacă afișează răspunsul corect și folosește maxim $2 \cdot N$ interogări pentru fiecare test.

A doua grupă (90 points)

- $N = 1\,000$
- $T = 300$
- $0.001 \leq P \leq 0.2$

Pe această grupă se pot obține punctaje parțiale.

Dacă răspunsul pe un scenariu este greșit, veți primi zero puncte. Altfel, numărul de puncte pe un test va fi determinat pe baza mediei dintre numărul de interogări pe fiecare scenariu. În general, un număr mai mic de interogări va conduce la număr mai mare de puncte. Fie Q numărul mediu de interogări folosite de program pe toate scenariile, rotunjit în jos la o singură zecimală. Pentru fiecare test, am calculat o valoare F (vezi mai jos). Scorul pe un test va fi calculat conform următoarelor reguli:

- Dacă $Q > 10 \cdot F$ vei primi 0 puncte (wrong answer).
- Dacă $F < Q \leq 10 \cdot F$, numărul de puncte se va calcula pe baza formulei:

$$90 \cdot \frac{F}{F + 4 \cdot (Q - F)}$$

- Dacă $Q \leq F$, vei obține 90 puncte.

Soluția ta va fi evaluată pe mai multe teste cu diferite valori ale lui P. Numărul total de puncte pe care îl vei obține va fi egal cu numărul minim de puncte obținut pe fiecare dintre teste (i.e., pentru toate probabilitățile P).

Sunt următoarele teste:

P	F
0.001	15.1
0.005256	51.1
0.011546	94.9
0.028545	191.5
0.039856	246.3
0.068648	366.2
0.104571	490.3
0.158765	639.1
0.2	731.4

Sistemul de evaluare va oferi feedback pentru fiecare test. Acest feedback va conține valoarea Q obținută de soluția ta pe fiecare test cu un scor nenul.

Exemplu de interacțiune

Nu uitați să dați flush la output după fiecare linie afișată. Țineți cont de faptul că următoarele valori pentru N și T nu pot să apară în niciunul dintre teste.

Input	Output
10 0.4 2	
	Q 1000000000
P	
	Q 0000001000
P	
	Q 0000000001
P	
	Q 0111110110
N	
	A 1000001001
C	
	A 0000000000
W	

Programul a rezolvat primul scenariu corect, dar nu și al doilea, deoarece soluția corectă era `1100010010` (pe care programul nu putea să o știe, deoarece nu a făcut nicio interogare). Chiar dacă ar mai fi fost alte scenarii de rezolvat, programul trebuia să se oprească imediat.