COVID testi

Na Adamovi šoli se je pojavil nov val epidemije COVID. Da bi preprečili nadaljnje širjenje, se je šola odločila, da bo vse učence testirala z antigenskimi testi z vzorci sline učencev. Ker so vsi učitelji že zdavnaj pozabili, kako uporabljati take teste, se je Adam javil kot prostovoljec za pomoč pri testiranju.

Prejel je vzorce sline od N učencev (zaradi zasebnosti lahko ve le identifikacijske številke od 0 do $N-1$), njegova naloga pa je, da določi, kateri vzorci so pozitivni. Iz lokalne statistike ve, da je verjetnost, da bo katerikoli študent pozitiven, enaka P, in da so testi med seboj neodvisni (za bolj formalen opis glej razdelek "Vhodni podatki"). Žal je prepozno ugotovil, da je testiranje vseh študentov izredno dolgo in dolgočasno opravilo. Na srečo je spoznal, da lahko postopek testiranja izvede na pametnejši način kot z zaporednim testiranjem vseh vzorcev. Če testira mešanico podmnožice vseh vzorcev, bo ugotovil, ali so vsi vzorci v tej mešanici negativni ali pa je vsaj eden od njih pozitiven. Take skupinske teste bi lahko uporabil za zmanjšanje števila testov, ki jih mora opraviti!

V vsakem vzorcu je dovolj sline, da lahko vzorec testira tolikokrat, kot želi. Poleg tega so testi povsem natančni, zato se nikoli ne zgodi, da bi različni testi pri isti osebi dali različne rezultate.

Adam bi rad optimiziral postopek tako, da bi uporabil čim manj testov. Ker pa je sam zaposlen s testiranjem, je optimizacija postopka odvisna od tebe.

Komunikacija

Ta naloga je interaktivna.

Tvoj program se bo izvajal na več testnih primerih. Za rešitev enega primera (tekom enega izvajanja programa) boste morali rešiti T različnih podprimerov. Vrednosti N in P sta pri vseh podprimerih enaki, vendar bodo učenci, ki so pozitivni, v vsakem podprimeru (najverjetneje) različni.

Komunikacijski protokol lahko implementirate sami ali pa uporabite predlogo, ki je na voljo v sistemu CMS kot priponka nalogi z imenom `template.cpp`.

Protokol

Vaš program mora najprej s standardnega vhoda prebrati vrstico, ki vsebuje naslednja s presledki ločena števila: celo število N (število učencev), decimalno število P (verjetnost pozitivnega vzorca) in celo število T (število podprimerov).

Nato lahko program začne z izpisom poizvedb na standardni izhod in reševati prvi podprimer. Vsaka poizvedba mora biti ena sama vrstica, ki vsebuje znak `Q`, presledek in opis mešanice slin. Opis mešanice predstavimo z nizom s dolžine N, kjer je i-ti znak `1`, če je slina i-tega učenca v mešanici, sicer pa `0`. Po izpisu te vrstice mora program poplakniti standardni izhod (`cout.flush()` oz. `fflush(stdout)`). Nato s standardnega vhoda prebereš odgovor sistema na to poizvedbo, ki je znak `P` za primer, da je okužena vsaj ena slina v mešanici, sicer pa `N` (vsi vzorci so negativni).

Ko ima program dovolj podatkov za rešitev podprimera, naj jo izpiše na standardni izhod v eni vrstici, ki vsebuje znak `A`, presledek in opis okuženosti učencev (torej i-ti znak je `1` za okuženega i-tega učenca, sicer pa je `0`). Po izpisu te vrstice mora program izprazniti standardni izhod (`flush`) in nato prebrati vrstico z odgovorom sistema.

- Če vrstica vsebuje le znak `C`, je bil tvoj odgovor pravilen. V tem primeru lahko program začne izvajati poizvedbe o naslednjem podprimeru, oziroma naj se zaključi, če je bil to zadnji (torej, T-ti) podprimer.
- Če vrstica vsebuje znak `W`, potem tvoj odgovor ni bil pravilen in program se mora takoj zaključiti. To je pomembno zato, da boš dobil pravilno povratno informacijo od CMS. Če se program ne zaključi, bo lahko ocenjen kot da se je sesul, trajal predolgo, ali kaj drugega.

Uporaba predloge `template.cpp`

Pri implementaciji protokola si lahko pomagaš s predlogami v `template.cpp`; v tem primeru moraš implementirati samo funkcijo `std::vector<bool> find_positive()`. Ta funkcija se bo za vsak podprimer poklicala enkrat. Vrniti mora seznam booleanov dolžine N, kjer je i-ti element `true`, če je i-ti učenec okužen.

Funkcijo `bool test_students(std::vector<bool> mask)` lahko uporabljate, da sistem vprašate za rezultat testa mešanice slin učencev. Edini argument funkcije je seznam booleanov dolžine N, kjer je i-ti element `true`, če je treba i-ti vzorec dodati v mešanico. Funckija vrne `true`, natanko tedaj, ko je vsaj eden od vzorcev v mešanici pozitiven.

Uporabite lahko tudi globalni spremenljivki `N` in `P`, ki vsebujeta vrednosti N in P. Po prvem klicu funkcije `scanf` lahko v funkciji `main` opravite kakršnokoli inicializacijo.

Vhodni podatki

Sodnik pri tej nalogi ni prilagodljiv, kar pomeni, da je okuženost posameznih učencev določena pred zagonom vašega programa. Ali je posamezen učenec zdravstveno oporečen, je določeno neodvisno z verjetnostjo P z uporabo poštenega generatorja naključnih števil.

Podnaloge in ocenjevanje

Naloga ima dve podnalogi.

Prva podnaloga (10 točk)

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

Rešitev dobi vseh 10 točk, če za vsak testni primer odgovori pravilno in pri tem naredi največ $2 \cdot N$ poizvedb. Sicer za to podnalogo dobi 0 točk.

Druga podnaloga (90 točk)

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

Ta podnaloga uporablja delno točkovanje. Na kratko:

- Število točk za celo podnalogo je minimum točk po posameznih testnih primerih.
- Vsak testni primer je točkovan glede na učinkovitost tvojega programa (torej da čez vse podprimere naredi skupno čim manj poizvedb); seveda morajo biti vsi podprimeri pri tem rešeni pravilno (sicer ne dobi nobene točke).

Natančneje:

Tvoja rešitev bo ocenjena na več testnih primerih z različnimi vrednostmi P. Skupno število točk, ki jih prejmeš za nalogo, bo enako najmanjšemu številu točk, ki ga dobiš pri posameznem testnem primeru (tj. minimum po vseh primerih za verjetnosti P).

Če je odgovor tvojega programa na katerikoli podprimer napačen, za to podnalogo ne dobiš nobene točke. V nasprotnem primeru bo število točk za vsak testni primer določeno na podlagi povprečnega števila poizvedb na podprimer. Na splošno velja, da manjše število poizvedb prinaša večje število točk. Naj Q označuje povprečno število poizvedb, ki jih naredi tvoj program v vseh podprimerih, zaokroženo navzdol na eno decimalno mesto. Za vsak testni primer so sestavljalci izračunali vrednost F (glej tabelo spodaj). Rezultat bo izračunan po naslednjih pravilih:

- Če $Q > 10 \cdot F$, prejmeš 0 točk (napačen odgovor).
- Če $F < Q \leq 10 \cdot F$, bo število točk določeno po naslednji formuli:

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

- Če je $Q \leq F$, prejmeš polnih 90 točk.

Podnalogo sestavljajo naslednji testni primeri (tj. za to podnalogo že poznaš N, P, T):

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

Ocenjevalni sistem bo vrnil povratne informacije za vsak testni primer. Za vsak primer, kjer je program dobil neničelno število točk, bo sistem prikazal število poizvedb Q, ki jih je program naredil.

Primer interakcije

Sledi primer interakcije z ocenjevalnim programom. Upoštevaj, da se spodnja N in T ne moreta pojaviti pri nobeni podnalogi. Po vsakem izpisu ne pozabi poplakniti za seboj (`flush`).

Tvoj vhod	Tvoj izhod
`10 0.4 2`	
	`Q 1000000000`
`P`	
	`Q 0000001000`
`P`	
	`Q 0000000001`
`P`	
	`Q 0111110110`
`N`	
	`A 1000001001`
`C`	
	`A 0000000000`
`W`	

Program je pravilno rešil prvi podprimer, ne pa tudi drugega, saj je bila pravilna rešitev `1100010010` (ki je program ni mogel poznati, saj ni opravil nobenih poizvedb). Tudi če bi obstajal še kakšen podprimer, se mora program takoj končati in ga ne poskušati reševati.