COVID Tests

In Adams Schule ist die COVID-Pandemie wieder ausgebrochen. Um weitere Ansteckungen zu verhindern, hat die Schule beschlossen, alle Schulkinder zu testen. Dafür sollen Antigen-Tests mit Speichelproben aller Schulkinder durchgeführt werden.

Da alle Lehrpersonen vor langer Zeit vergessen haben, wie man diese benutzt, hat Adam sich freiwillig als Helfer beim Testen gemeldet.

Er erhält Speichelproben von N Schulkindern (aus Datenschutzgründen kennt er nur ihre Kennziffern von 0 bis $N-1$) und muss ermitteln, welche dieser Proben positiv sind. Leider realisiert er zu spät, wie langweilig und mühsam es ist, all diese Proben auszuwerten. Allerdings fällt ihm auf, dass er die Proben nicht eine nach der anderen bearbeiten muss. Wenn er stattdessen eine Teilmenge der Proben auswählt, den enthaltenen Speichel vermischt und die Mischung testet, kann er bestimmen, ob alle Proben in der Mischung negativ sind oder ob mindestens eine davon positiv ist. Damit könnte er die Anzahl der benötigten Tests reduzieren!

Weil jede Probe ausreichend viel Speichel enthält, kann er jede einzelne Probe beliebig oft testen. Ausserdem sind die Tests unfehlbar. Es kommt also nie vor, dass unterschiedliche Auswertungen derselben Probe zu unterschiedlichen Ergebnissen führen.

Unter diesen Voraussetzungen würde er den Prozess gerne so optimieren, dass er so wenig Tests wie möglich durchführen muss. Leider ist er zu sehr mit dem Testen beschäftigt. Die Prozessoptimierung ist also deine Aufgabe.

Laut örtlicher Statistik ist eine Probe mit Wahrscheinlichkeit P positiv. Ausserdem hat das positive oder negative Ergebnis einer Probe keinerlei Einfluss auf andere Proben. Vielleicht hilft dir das beim Optimieren seiner Teststrategie?

Kommunikation

Dies ist eine interaktive Aufgabe.

Dein Programm wird auf mehreren Testfällen ausgeführt. Innerhalb eines Testfalls, also während eines Durchlaufs deines Programms, musst du T verschiedene Szenarien bearbeiten. In allen T Szenarien haben N und P den gleichen Wert. Höchstwahrscheinlich sind aber nicht die gleichen Proben positiv.

Du kannst das folgende Protokoll entweder selbst implementieren oder das bereitgestellte Template nutzen. Dieses kannst du im CMS als Anhang zur Aufgabe unter dem Namen `template.cpp` finden.

Protokoll

Zuerst soll dein Programm eine Zeile von der Standardeingabe einlesen. Diese beinhaltet eine ganze Zahl N, eine reelle Zahl P und eine ganze Zahl T, die alle durch Leerzeichen voneinander getrennt sind. Diese sind die Anzahl der Schulkinder, die Wahrscheinlichkeit einer positiven Probe und die Anzahl der Szenarien.

Danach kann dein Programm Anfragen auf der Standardausgabe ausgeben. Eine Anfrage soll aus einer einzelnen Zeile bestehen: `Q`, ein Leerzeichen und eine Zeichenkette s mit Länge N, wobei s_i den Wert `1` hat, falls Adam die Probe des i-ten Schulkindes zum Testpool hinzufügen soll, und sonst `0`. Nach der Ausgabe dieser Zeile muss dein Programm die Standardausgabe flushen und eine einzige Zeile mit einem einzelnen Buchstaben einlesen. Dieser Buchstabe ist `P`, falls mindestens ein Schulkind aus dem Testpool positiv ist, und sonst `N` .

Die Ausgabe der Antwort besteht ebenfalls aus einer einzelnen Zeile: `A`, ein Leerzeichen und eine Zeichenkette s der Länge N, in der s_i den Wert 1 hat, falls das `i`-te Schulkind positiv ist und `0` sonst. Nach der Ausgabe dieser Zeile muss dein Programm die Standardausgabe flushen und dann einen einzelnen Buchstaben in einer einzelnen Zeile einlesen.

Falls dieser Buchstabe `C` ist, war deine Antwort korrekt. In diesem Fall kann dein Programm anfangen, Anfragen für das nächste Szenario zu stellen oder sich selbst beenden, wenn dies bereits die T-te Antwort war.

Falls dieser Buchstaben `W` ist, war deine Antwort falsch. In diesem Fall soll sich dein Programm sofort beenden.

Dieses Abbrechen des Programms ist wichtig, um die richtige Rückmenldung vom CMS zu erhalten. Falls dein Programm trotz `W` weiterläuft, kann es abstürzen oder eine andere Fehlermeldung zurückgeben.

Das Template

Wenn du die bereitgestellte Implementierung des Protokolls aus `template.cpp` benutzt, musst du die Funktion `std::vector<bool> find_positive()` implementieren. Diese Funktion wird einmal für jedes Szenario aufgerufen. Sie muss einen Vector der Länge N mit Boolschen Einträgen zurückgeben, wobei das i-te Element genau dann `true` ist, wenn die Probe des i-ten Schulkindes positiv ist.

Um dein Verfahren zu implementieren, kannst du die Funktion `bool test_students(std::vector<bool> mask)` verwenden. Diese Funktion testet eine Teilmenge von Proben. Sie nimmt ein einziges Argument entgegen, und zwar einen Vector der Länge N mit Boolschen Einträgen, in dem das i-te Element genau dann `true` ist, wenn es zum Testpool hinzugefügt werden soll. Sie gibt dann und nur dann `true` zurück, wenn mindestens eine der Proben im Testpool positiv ist.

Du kannst auch die globalen Variabeln `N` und `P` benutzen, die N und P enthalten. Nach dem ersten Aufruf von `scanf` darfst du Initialisierungen innerhalb der `main`-Funktion vornehmen.

Eingabe

In dieser Aufgabe ist der Grader nicht adaptiv. Das bedeutet, dass vor der Ausführung des Programms festgelegt wird, welche Proben positiv sind und welche nicht. Dies wird für alle Proben unabhängig voneinander mithilfe eines fairen Zufallszahlengenerators mit Wahrscheinlichkeit P bestimmt.

Teilaufgaben und Bewertung

Es gibt zwei Teilaufgaben.

Erste Teilaufgabe (10 Punkte)

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

Dine Lösung wird akzeptiert, wenn sie die korrekte Antwort liefert und pro Szenario höchstens $2 \cdot N$ Anfragen stellt.

Zweite Teilaufgabe (90 Punkte)

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

Diese Teilaufgabe verwendet Partial Scoring.

Falls deine Antwort in irgendeinem Szenario nicht korrekt ist, erhälst du gar keine Punkte. Andernfalls wird deine Punktzahl für einen beliebigen Testfall basierend auf der durchschnittlichen Anzahl von Anfragen pro Szenario bestimmt. Im Allgemeinen ergibt eine geringere Anzahl von Anfragen eine höhere Punktzahl. Sei Q die durchschnittliche Anzahl von Anfragen nach Abrunden

auf eine Dezimalstelle. Für jeden einzelnen Testfall haben wir einen Wert F berechnet (siehe unten). Die Punktzahl eines gegebenen Testfalls wird nach den folgenden Regeln berechnet:

- Falls $Q > 10 \cdot F$, erhältst du 0 Punkte (wrong answer).
- Falls $F < Q \leq 10 \cdot F$, wird die Punktzahl mit folgender Formel bestimmt:

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

- Falls $Q \leq F$, erhältst du alle 90 Punkte.

Deine Lösung wird auf mehreren Testfällen mit verschiedenen Werten für P getestet. Die resultierende Gesamtpunktzahl ist die kleinste Punktzahl unter allen Testfällen (mit der gleichen Wahrscheinlichkeit P).

Es gibt die folgenden Testfälle:

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

Der Grader liefert eine Rückmeldung für jeden einzelnen Testfall. Diese Rückmeldung enthält den Wert Q deiner Lösung zu jedem Testfall, auf dem du mehr als 0 Punkte erhältst.

Beispiel für eine Interaktion

Hier ist eine Beispielinteraktion mit dem Grader. Dabei gehören diese Werte von N und T zu keiner der Teilaufgaben. Vergiss nicht, die Standardausgabe nach der Ausgabe einer jeden einzelnen Zeile zu flushen.

Deine Eingabe	Deine Ausgabe
`10 0.4 2`	
	`Q 1000000000`
`P`	
	`Q 0000001000`
`P`	
	`Q 0000000001`
`P`	
	`Q 0111110110`
`N`	
	`A 1000001001`
`C`	
	`A 0000000000`
`W`	

Das Programm hat das erste Szenario korrekt gelöst, das zweite aber nicht. Die richtige Lösung war nämlich `1100010010`. (Dies konnte das Programm nicht wissen, weil es keine Anfragen gestellt hat). Auch wenn es ein weiteres Szenario gäbe, müsste das Programm sofort abbrechen.