

Elevator

Time limit: 10 seconds

Memory limit: 64 MiB (per process)

Aceasta este o problemă de comunicare.

Există o clădire cu etaje numerotate de la 0 la N . Etajul 0 este parter, iar deasupra etajului N se află acoperișul. Astfel, inclusiv acoperișul, clădirea are $N + 2$ niveluri.

Fiecare etaj f ($0 \leq f \leq N$) este ocupat de exact un locuitor și are un număr secret v_f ($0 \leq v_f \leq 3$), cunoscut numai de către locuitorul său. De asemenea, pe acoperișul clădirii locuiește Karlsson. Obiectivul este ca Karlsson să determine cât mai multe dintre valorile $v_0, v_1, \dots, v_N$ posibil, iar fiecare locuitor **cooperează** pentru a maximiza numărul de valori pe care acesta le poate recupera.

Pentru a comunica, locuitorii folosesc un ascensor al clădirii în felul următor.

Ascensorul începe la etajul 0 și se deplasează doar în sus. În interiorul ascensorului sunt butoane etichetate cu etajele de la 1 până la N . Inițial, niciun buton nu este apăsat. Odată ce un buton este apăsat, rămâne apăsat permanent. Ascensorul se oprește și se deschide la etajul f dacă fie $f = 0$, fie butonul corespunzător etajului f a fost apăsat la o oprire anterioară. După vizitarea celui mai înalt etaj al cărui buton a fost apăsat (sau etajul 0 dacă niciun buton nu este apăsat vreodată), ascensorul continuă direct către terasă fără a se opri la niciunul dintre etajele rămase.

Când ascensorul se oprește la etajul f se întâmplă următoarele:

1. Locuitorul vede mulțimea completă de butoane care sunt apăstate la momentul curent.
2. Bazat doar pe această informație și pe valoarea v_f , poate alege orice submulțime din butoanele neapăsate corespunzând etajelor **strict deasupra** propriului etaj. Locuitorul apasă aceste butoane.
3. Ascensorul urcă la următorul etaj cu numărul cel mai mic pentru care butonul corespunzător a fost apăsat (sau la terasă dacă au fost vizitate toate etajele care corespund butoanelor apăstate).

The elevator starts on floor 0 and travels only upwards. Inside the elevator there are buttons labelled with floors 1 through N . Initially, no buttons are pressed. Once a button is pressed, it remains pressed permanently. The elevator stops and opens on floor f if either $f = 0$ or the button corresponding to floor f has been pressed at an earlier stop. After visiting the highest floor whose button has been pressed (or floor 0 if no buttons are ever pressed), the elevator continues directly to the rooftop without stopping at any remaining floors.

Dacă liftul nu se oprește la un anumit etaj, locuitorul corespunzător nu poate apăsa niciun buton.

Când liftul ajunge la acoperiș, Karlsson vede doar mulțimea finală a butoanelor apăstate. Folosind această informație, el încearcă să recupereze cât mai multe dintre valorile $v_0, v_1, \dots, v_N$ posibil.

Valorile v sunt fixate înainte ca programul tău să înceapă și nu se schimbă în timpul procesului.

Detalii de implementare

Vor fi T cazuri de test.

Trimiteți un fișier care implementează aceste două funcții.

Prima funcție pe care ar trebui să o implementați este următoarea:

```
std::vector<int> press_buttons(int subtask, int N,  
                              int f, int v, std::vector<int> p)
```

- $subtask$: subtask-ul pentru care această funcție este apelată, unde $0 \leq subtask \leq 4$;
- N : numărul ultimului etaj;
- f : etajul curent, unde $0 \leq f \leq N$;
- v : valoarea v_f scrisă la etajul f ;
- p : butoanele apăstate la momentul curent, în ordine crescătoare.

Această funcție este apelată când ascensorul se deschide la etajul f (dacă fie $f = 0$, fie butonul corespunzător etajului f a fost apăsat la o oprire anterioară). Această funcție nu va fi apelată pentru un anumit etaj dacă butonul corespunzător nu a fost apăsat anterior.

Valoarea returnată este lista butoanelor nou apăstate. Ordinea valorilor în tabloul returnat nu este importantă. Fiecare buton apăsat x trebuie să satisfacă următoarele:

- $f < x \leq N$;
- x nu se află deja în p și x apare exact o dată în tabloul returnat.

A doua funcție pe care trebuie să o implementați este următoarea:

```
std::vector<int> answer(int subtask, int N, std::vector<int> p)
```

- $subtask$: subtask-ul pentru care această funcție este apelată unde, $0 \leq subtask \leq 4$;
- N : numărul ultimului etaj;
- p : mulțimea finală de butoane apăstate, în ordine crescătoare.

Această funcție este apelată exact o dată pentru fiecare test când ascensorul ajunge la acoperiș.

Tabloul returnat trebuie să aibă lungimea de $N + 1$, al i -lea element din tablou ar trebui să fie egal cu v_i , în timp ce valorile pe care Karlsson **nu poate să le recupereze** ar trebui să fie înlocuite cu -1 .

Important: Persoanele din clădire nu pot comunica în niciun alt mod decât prin butoanele apăsate. Pentru a asigura acest lucru, programul dvs. va fi executat ca $N + 2$ procese separate, câte unul pentru fiecare etaj și unul pentru acoperiș. În fiecare proces pentru etaje, vor fi cel mult T apeluri ale funcției `press_buttons` și în procesul pentru acoperiș – exact T apeluri ale funcției `answer`. Prin urmare, programul dvs. nu ar trebui să presupună că există nicio stare partajată (de exemplu variabile globale). Fiecare proces va avea 64 MiB de memorie disponibilă care este separată de memoria altor procese, iar toate aceste apeluri (până la $(N + 1) \times T_{\text{press_buttons}}$ și exact T_{answer}) trebuie să se finalizeze în 10 secunde.

Restricții

- $N = 60$
- $T \leq 10\,000$
- $0 \leq v_i \leq 3$ pentru orice $0 \leq i \leq N$

Exemple

Testul exemplu test are 1 test case cu următoarele valori ale etajelor:

```
1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 0 1 1 1
1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 0 1 1
```

Iată un exemplu de interacțiune

Participant program	Jury program
	<code>press_buttons(0, 60, 0, 1, {})</code>
<code>return {2}</code>	
	<code>press_buttons(0, 60, 2, 0, {2})</code>
<code>return {13, 42}</code>	
	<code>press_buttons(0, 60, 13, 0, {2, 13, 42})</code>
<code>return {}</code>	
	<code>press_buttons(0, 60, 42, 1, {2, 13, 42})</code>
<code>return {}</code>	
	<code>answer(0, 60, {2, 13, 42})</code>
<code>return {1, -1, 0, -1, -1, ..., -1}</code>	

Această interacțiune recuperează corect valorile 0 și 2. Celelalte valori returnate sunt egale cu -1 deoarece Karlsson nu le-a recuperat.

Subtask-uri

Subtask	Puncte	Restricții suplimentare	Numărul de etaje de recuperat pentru punctaj complet
0	0	Exemplul.	–
1	15	$0 \leq v_i \leq 1$ $v_0 = v_N = 1$ Dacă $v_i = 0$, atunci $v_{i+1} = 1$ pentru orice $0 \leq i < N$	60
2	35	$0 \leq v_i \leq 1$	40
3	15	$0 \leq v_i \leq 2$	30
4	35	$0 \leq v_i \leq 3$	25

Punctaj

Dacă programul tău returnează o valoare recuperată incorect sau efectuează o operație nevalidă (de exemplu returnează un `vector` de lungime greșită) în oricare din cazurile de testare pentru un subtask, submisia ta va primi zero puncte pentru acel subtask.

Pentru un caz de testare, *numărul de recuperări* al submisiei tale este numărul de poziții a căror valoare returnată nu este -1 . Fie K cel mai mic număr de recuperări dintre toate cazurile de testare ale unui subtask (fiecare subtask are doar 1 test care conține până la 10 000 cazuri de testare). Apoi, punctele pentru fiecare subtask sunt calculate conform tabelului de mai jos.

Subtask	Domeniul pentru K	Puncte
0	-	0
1	$K < 60$	$0.25K$
	$K \geq 60$	15
2	$K < 30$	$0.5K$
	$30 \leq K < 40$	$2K - 45$
	$K \geq 40$	35
3	$K < 30$	$0.5K$
	$K \geq 30$	15
4	$K < 20$	$0.7K$
	$20 \leq K < 25$	$4K - 66$
	$K \geq 25$	35

Sample grader

Sample grader efectuează toate apelurile de funcții pentru toate cazurile de test într-o singură execuție.

Formatul de intrare este următorul:

- linia 1: trei întregi – numărul de cazuri de test T , numărul ultimului etaj N și numărul subtaskului S ;
- linia $1 + i$: $N + 1$ întregi $v_0, v_1, \dots, v_N$, unde v_f este valoarea scrisă pe etajul f pentru cazul de test.

Formatul de ieșire este următorul:

- linia i : un întreg – numărul de valori ale v corect recuperate în cazul de test i ;

- linia $T + 1$: punctajul final.

Pentru feedback mai detaliat, schimbă macro-ul `DETAILED` din `false` în `true` pe prima linie a graderului.

Poți lăsa evaluatorul să genereze cazuri de test (valori ale v_f) pentru tine schimbând macro-ul `AUTO_GENERATE` din `false` în `true` pe a doua linie a graderului.