

Winda

Limit czasu: 10 sekund

Limit pamięci: 64 MiB (na proces)

To jest zadanie komunikacyjne.

Budynek ma piętra ponumerowane od 0 do N . Piętro 0 to parter, a nad piętrem N znajduje się dach. Zatem, wliczając dach, budynek ma $N + 2$ poziomy.

Każde piętro f ($0 \leq f \leq N$) jest zajmowane przez dokładnie jednego mieszkańca i posiada tajną liczbę całkowitą v_f ($0 \leq v_f \leq 3$), znaną tylko przez niego. Na dachu tego budynku mieszka prezes Rafał. Wszyscy mieszkańcy **współpracują**, aby Rafał odgadł jak największą wartość $v_0, v_1, \dots, v_N$.

W celu komunikacji mieszkańcy korzystają z windy w budynku w następujący sposób.

Winda startuje z piętra 0 i porusza się tylko w górę. Wewnątrz windy znajdują się przyciski oznaczone 1 do N odpowiadające piętrům. Na początku żaden przycisk nie jest wciśnięty. Gdy przycisk zostanie wciśnięty, nigdy już nie zgaśnie. Winda zatrzymuje się i otwierają się drzwi na piętrze f , jeśli $f = 0$ lub przycisk odpowiadający piętru f został kiedyś wcześniej wciśnięty. Po odwiedzeniu najwyższego piętra, którego przycisk został wciśnięty (lub piętra 0, jeśli żaden przycisk nigdy nie zostanie wciśnięty), winda jedzie bezpośrednio na dach bez zatrzymywania się na pozostałych piętrach.

Gdy winda zatrzymuje się na piętrze f :

1. Mieszkaniec widzi wszystkie obecnie wciśnięte przyciski.
2. Opierając się wyłącznie na tej informacji i wartości v_f , może wybrać dowolny podzbiór aktualnie niewciśniętych przycisków odpowiadających piętrům znajdującym się **ściśle powyżej** jego własnego piętra. Mieszkaniec wciska wybrane przyciski.
3. Winda jedzie do następnego najniższego piętra, które jest wybrane (lub na dach, jeśli wszystkie wybrane piętra zostały już odwiedzone).

Jeśli winda nie zatrzymuje się na pewnym piętrze, jego mieszkaniec nie może wcisnąć żadnych przycisków.

Gdy winda dotrze na dach, Rafał zobaczy końcowy zbiór wciśniętych przycisków. Korzystając z tej informacji, próbuje on odzyskać jak największą wartość $v_0, v_1, \dots, v_N$.

Wartości v są ustalone przed uruchomieniem Twojego programu i nie ulegają zmianie w trakcie jego działania.

Szczegóły implementacji

Należy rozwiązać T przypadków testowych.

Prześlij jeden plik implementujący dwie funkcje opisane poniżej.

Pierwsza funkcja, którą powinienes zaimplementować, jest następująca:

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

- $subtask$: podzadanie, dla którego wywoływana jest ta funkcja, gdzie $0 \leq subtask \leq 4$;
- N : numer ostatniego piętra;
- f : obecne piętro, gdzie $0 \leq f \leq N$;
- v : wartość v_f zapisana na piętrze f ;
- p : obecnie wciśnięte przyciski w kolejności rosnącej.

Ta funkcja jest wywoływana, gdy winda zatrzyma się na piętrze f (jeśli $f = 0$ albo przycisk odpowiadający temu piętru został wcześniej wybrany). Ta funkcja nie zostanie wywołana dla danego piętra, jeśli przycisk mu odpowiadający nie został wcześniej wybrany.

Wartością zwracaną jest lista nowych przycisków do wciśnięcia. Kolejność wartości w zwracanej tablicy nie ma znaczenia. Każdy wciśnięty przycisk x musi spełniać następujące warunki:

- $f < x \leq N$;
- x nie znajduje się jeszcze w p i pojawia się dokładnie raz w zwracanej tablicy.

Druga funkcja, którą powinienes zaimplementować, to:

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

- $subtask$: podzadanie, dla którego wywoływana jest ta funkcja, gdzie $0 \leq subtask \leq 4$;
- N : numer ostatniego piętra;
- p : końcowy zbiór wciśniętych przycisków w kolejności rosnącej.

Ta funkcja jest wywoływana dokładnie raz na przypadek testowy, gdy winda dotrze na dach.

Zwracana tablica musi mieć długość $N + 1$, jej i -ty element powinien być równy v_i albo -1 , jeśli Rafał jej **nie odzyskał**.

Ważne: Osoby w budynku nie mogą się komunikować w żaden inny sposób, niż poprzez wciśnięte przyciski. Aby to zapewnić, Twój program zostanie uruchomiony jako $N + 2$ oddzielnych procesów,

po jednym dla każdego piętra i jednym dla dachu. W każdym procesie dla pięter nastąpi co najwyżej T wywołań funkcji `press_buttons`, a w procesie dla dachu – dokładnie T wywołań funkcji `answer`. Dlatego Twój program nie może korzystać z żadnego współdzielonego stanu (na przykład, zmiennych globalnych). Każdy proces będzie miał przydzielone do dyspozycji 64 MiB pamięci, niezależnej od pamięci innych procesów. Wszystkie wywołania (do $(N + 1) \times T$ `press_buttons` i dokładnie T `answer`) muszą zakończyć się w ciągu 10 sekund.

Ograniczenia

- $N = 60$
- $T \leq 10\,000$
- $0 \leq v_i \leq 3$ dla każdego $0 \leq i \leq N$

Przykład

Przykładowy test zawiera 1 przypadek testowy z następującymi wartościami pięter:

```
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 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 1 0 1 1
```

Oto przykład interakcji:

Program uczestnika	Program jury
	<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>	

Ta interakcja poprawnie odzyskuje wartości o indeksach 0 i 2. Wszystkie pozostałe zwrócone wartości są równe -1 , ponieważ Rafał ich nie odzyskał.

Podzadania

Podzadanie	Punkty	Dodatkowe ograniczenia	Liczba pięter do odzyskania dla pełnych punktów
0	0	Przykład z zadania.	–
1	15	$0 \leq v_i \leq 1$ $v_0 = v_N = 1$ Jeśli $v_i = 0$ wtedy $v_{i+1} = 1$ dla każdego $0 \leq i < N$	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

Ocenianie

Jeśli Twój program zwróci niepoprawnie odzyskaną wartość lub wykona nieprawidłową operację (przykładowo zwróci `vector` o niewłaściwej długości) w którymkolwiek z przypadków testowych dla podzadania, Twoje zgłoszenie otrzyma zero punktów za to podzadanie.

Dla przypadku testowego *liczba odzyskanych wartości* w Twoim zgłoszeniu to liczba pozycji, których zwrócona wartość nie jest równa -1 . Niech K będzie minimalną liczbą odzyskanych wartości wśród wszystkich przypadków testowych w podzadaniu (każde podzadanie ma tylko 1 test zawierający do 10 000 przypadków testowych). Punkty za każde podzadanie są obliczane zgodnie z poniższą tabelą.

Podzadanie	Zakres K	Punkty
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

Przykładowy program ocenający

Przykładowy program ocenający wykonuje wszystkie wywołania funkcji dla wszystkich przypadków testowych w jednym procesie.

Format wejścia jest następujący:

- wiersz 1: trzy liczby całkowite – liczba przypadków testowych T , numer ostatniego piętra N oraz numer podzadania S ;
- wiersz $1 + i$: $N + 1$ liczb całkowitych $v_0, v_1, \dots, v_N$, gdzie v_f to wartość zapisana na piętrze f dla przypadku testowego.

Format wyjścia jest następujący:

- wiersz i : jedna liczba całkowita – liczba poprawnie odzyskanych wartości v w przypadku testowym i ;
- wiersz $T + 1$: końcowe punkty.

Aby uzyskać bardziej szczegółowe informacje zwrotne, zmień makro `DETAILED` z `false` na `true` w pierwszej linii kodu źródłowego programu ocenającego.

Możesz pozwolić sprawdzarce generować przypadki testowe (wartości v_f) dla Ciebie, zmieniając makro `AUTO_GENERATE` z `false` na `true` w drugiej linii kodu źródłowego programu ocenającego.