

Ліфт

Обмеження часу: 10 секунд

Обмеження пам'яті: 64 MiB (на процес)

Це задача на комунікацію.

У мальовничому місті Києві, в легендарному районі Троєщина, налічується близько 2000 багатоповерхових будинків. Є будинок з поверхами, пронумерованими від 0 до N . Поверх 0 є першим поверхом, а над поверхом N розташований дах. Отже, разом із дахом будинок має $N + 2$ рівні.

На кожному поверсі f ($0 \leq f \leq N$) живе рівно один мешканець, і на ньому записане секретне ціле число v_f ($0 \leq v_f \leq 3$), відоме лише цьому мешканцю. Також на даху цього будинку живе Карлсон. Мета полягає в тому, щоб Карлсон визначив якомога більше значень $v_0, v_1, \dots, v_N$, причому кожен мешканець **співпрацює**, щоб максимізувати кількість значень, які Карлсон зможе відновити.

Для комунікації мешканці використовують ліфт будинку в такий спосіб.

Ліфт починає рух із поверху 0 і рухається лише вгору. Усередині ліфта є кнопки, позначені номерами поверхів від 1 до N . Спочатку жодна кнопка не натиснута. Щойно кнопку натиснуто, вона назавжди залишається натиснутою. Ліфт зупиняється та відчиняє двері на поверсі f , якщо або $f = 0$, або кнопку, що відповідає поверху f , було натиснуто під час однієї з попередніх зупинок. Після відвідування найвищого поверху, кнопку якого було натиснуто (або поверху 0, якщо жодну кнопку так і не було натиснуто), ліфт без зупинок їде прямо на дах.

Коли ліфт зупиняється на поверсі f , відбувається таке:

1. Мешканець бачить повний набір кнопок, які наразі натиснуті.
2. Використовуючи лише цю інформацію та значення v_f , він може вибрати будь-яку підмножину ще не натиснутих кнопок, що відповідають поверхам **строго вище** за його власний поверх. Мешканець натискає ці кнопки.
3. Ліфт їде вгору до найближчого наступного поверху, відповідну кнопку якого було натиснуто (або на дах, якщо всі поверхи, що відповідають натиснутим кнопкам, уже було відвідано).

Якщо ліфт не зупиняється на деякому поверсі, відповідний мешканець не може натискати жодних кнопок.

Коли ліфт досягає даху, Карлсон бачить лише остаточний набір натиснутих кнопок. Використовуючи цю інформацію, він намагається відновити якомога більше значень $v_0, v_1, \dots, v_N$.

Значення v фіксуються до початку роботи вашої програми й не змінюються протягом усього процесу.

Деталі реалізації

Буде T тестових випадків.

Вам потрібно надіслати один файл, який реалізує дві функції.

Перша функція, яку потрібно реалізувати:

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

- $subtask$: номер підзадачі, для якої викликається ця функція, де $0 \leq subtask \leq 4$;
- N : номер останнього поверху;
- f : поточний поверх, де $0 \leq f \leq N$;
- v : значення v_f , записане на поверсі f ;
- p : поточні натиснуті кнопки у порядку зростання.

Ця функція викликається, коли ліфт відчиняє двері на поверсі f (якщо або $f = 0$, або кнопку, що відповідає поверху f , було натиснуто під час однієї з попередніх зупинок). Ця функція не викликатиметься для певного поверху, якщо відповідну кнопку не було натиснуто заздалегідь.

Поверненим значенням є список нових кнопок, які потрібно натиснути. Порядок значень у поверненому масиві не має значення. Кожна натиснута кнопка x повинна задовольняти такі умови:

- $f < x \leq N$;
- x ще не міститься в p і зустрічається в поверненому масиві рівно один раз.

Друга функція, яку потрібно реалізувати:

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

- $subtask$: номер підзадачі, для якої викликається ця функція, де $0 \leq subtask \leq 4$;
- N : номер останнього поверху;
- p : остаточний набір натиснутих кнопок у порядку зростання.

Ця функція викликається рівно один раз для кожного тестового випадку, коли ліфт досягає даху.

Повернений масив повинен мати довжину $N + 1$, його i -й елемент повинен дорівнювати v_i , а значення, які Карлсон **не може відновити**, потрібно замінити на -1 .

Важливо: Люди в будинку не можуть комунікувати жодним іншим способом, окрім натискання кнопок. Щоб це гарантувати, вашу програму буде запущено як $N + 2$ окремі процеси: по одному для кожного поверху та один для даху. У кожному процесі, що відповідає поверху, буде не більше ніж T викликів функції `press_buttons`, а в процесі, що відповідає даху, — рівно T викликів функції `answer`. Отже, ваша програма не повинна покладатися на будь-який спільний стан (наприклад, глобальні змінні). Кожен процес матиме 64 MiB пам'яті, окремої від пам'яті інших процесів, і всі ці виклики (до $(N + 1) \times T$ викликів `press_buttons` та рівно T викликів `answer`) повинні завершитися за 10 секунд.

Обмеження

- $N = 60$
- $T \leq 10\,000$
- $0 \leq v_i \leq 3$ для кожного $0 \leq i \leq N$

Приклад

У прикладі є 1 тестовий випадок із такими значеннями на поверхах:

```
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 1 0 1 1
```

Ось приклад взаємодії:

Програма учасника	Програма журі
	<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>	

У цій взаємодії правильно відновлено 0-ве та 2-ге значення. Усі інші повернені значення дорівнюють -1 , оскільки Карлсон не зміг їх відновити.

Підзадачі

Підзадача	Бали	Додаткові обмеження	Кількість поверхів, які потрібно відновити для повного бала
0	0	Приклад.	–
1	15	$0 \leq v_i \leq 1$ $v_0 = v_N = 1$ Якщо $v_i = 0$, то $v_{i+1} = 1$ для кожного $0 \leq i < N$	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

Оцінювання

Якщо ваша програма поверне неправильно відновлене значення або виконає некоректну операцію (наприклад, поверне `vector` неправильної довжини) у будь-якому тестовому випадку підзадачі, ваше рішення отримає нуль балів за цю підзадачу.

Для тестового випадку *кількість відновлених значень* вашого рішення — це кількість позицій, повернене значення яких не дорівнює -1 . Нехай K — мінімальна кількість відновлених значень серед усіх тестових випадків підзадачі (кожна підзадача містить лише 1 тест, що складається не більш ніж із 10 000 тестових випадків). Тоді кількість балів за кожну підзадачу обчислюється відповідно до таблиці нижче.

Підзадача	Діапазон K	Бали
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

Приклад градера

Приклад градера виконує всі виклики функцій для всіх тестових випадків у межах одного запуску.

Формат вхідних даних:

- рядок 1: три цілі числа — кількість тестових випадків T , номер останнього поверху N і номер підзадачі S ;
- рядок $1 + i$: $N + 1$ цілих чисел $v_0, v_1, \dots, v_N$, де v_f — значення, записане на поверсі f у цьому тестовому випадку.

Формат вихідних даних:

- рядок i : одне ціле число — кількість правильно відновлених значень v у тестовому випадку i ;
- рядок $T + 1$: підсумкові бали.

Для детальнішого зворотного зв'язку змініть макрос `DETAILED` з `false` на `true` у першому рядку градера.

Ви можете дозволити градеру генерувати тестові випадки (значення v_f), змінивши макрос `AUTO_GENERATE` з `false` на `true` у другому рядку градера.