

Elevator

Time limit: 10 seconds

Memory limit: 64 MiB (per process)

Это задача на коммуникацию.

Есть здание с этажами, пронумерованными от 0 до N . Этаж 0 — нижний этаж, а над этажом N находится крыша. Таким образом, включая крышу, в здании имеется $N + 2$ уровня.

На каждом этаже f ($0 \leq f \leq N$) живёт ровно один житель, и у него есть секретное целое число v_f ($0 \leq v_f \leq 3$), известное только ему. Также на крыше этого здания живёт Карлссон. Цель Карлссона — определить как можно больше значений $v_0, v_1, \dots, v_N$, при этом каждый житель **сотрудничает**, чтобы максимизировать количество значений, которые Карлссон сможет восстановить.

Для общения жители используют лифт здания следующим образом.

Лифт начинает движение с этажа 0 и движется только вверх. Внутри лифта находятся кнопки с номерами этажей от 1 до N . Изначально ни одна кнопка не нажата. После нажатия кнопка остаётся нажатой навсегда. Лифт останавливается и открывает двери на этаже f , если выполняется хотя бы одно из условий: $f = 0$ или кнопка, соответствующая этажу f , была нажата на одной из предыдущих остановок. После посещения самого высокого этажа, кнопка которого была нажата (или этажа 0, если ни одна кнопка никогда не нажималась), лифт продолжает движение непосредственно к крыше, не останавливаясь на остальных этажах.

Когда лифт останавливается на этаже f , происходит следующее:

1. Житель видит полный набор кнопок, которые в данный момент нажаты.
2. Основываясь только на этой информации и на значении v_f , он может выбрать любое подмножество ещё не нажатых кнопок, соответствующих этажам, находящимся **строго выше** его этажа. Житель нажимает эти кнопки.
3. Лифт поднимается к следующему ближайшему этажу, для которого соответствующая кнопка была нажата (или к крыше, если все этажи, соответствующие нажатым кнопкам, уже были посещены).

Если лифт не останавливается на каком-либо этаже, соответствующий житель не может нажать ни одной кнопки.

Когда лифт достигает крыши, Карлссон видит только итоговый набор нажатых кнопок. Используя эту информацию, он пытается восстановить как можно больше значений $v_0, v_1, \dots, v_N$.

Значения v фиксированы до запуска вашей программы и не изменяются во время процесса.

Implementation details

Будет T тестовых случаев.

Отправьте один файл, реализующий следующие две функции.

Первая функция, которую необходимо реализовать:

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

- $subtask$: подзадача, для которой вызывается эта функция, где $0 \leq subtask \leq 4$;
- N : номер последнего этажа;
- f : текущий этаж, где $0 \leq f \leq N$;
- v : значение v_f , записанное на этаже f ;
- p : текущие нажатые кнопки в порядке возрастания.

Эта функция вызывается, когда лифт открывается на этаже f (если $f = 0$ или кнопка, соответствующая этажу f , была нажата на одной из предыдущих остановок). Эта функция не будет вызываться для конкретного этажа, если соответствующая кнопка не была нажата ранее.

Возвращаемое значение — список новых кнопок, которые следует нажать. Порядок значений в возвращаемом массиве неважен. Каждая нажатая кнопка x должна удовлетворять следующим условиям:

- $f < x \leq N$;
- x ещё не содержится в p , и x встречается в возвращаемом массиве ровно один раз.

Вторая функция, которую необходимо реализовать:

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

- $subtask$: подзадача, для которой вызывается эта функция, где $0 \leq subtask \leq 4$;
- N : номер последнего этажа;
- p : итоговый набор нажатых кнопок в порядке возрастания.

Эта функция вызывается ровно один раз для каждого тестового случая, когда лифт достигает крыши.

Возвращаемый массив должен иметь длину $N + 1$, а его i -й элемент должен быть равен v_i ; значения, которые Карлссон **не может восстановить**, должны быть заменены на -1 .

Важно: жители здания не могут общаться никаким другим способом, кроме как через нажатые кнопки. Чтобы это обеспечить, ваша программа будет запущена как $N + 2$ отдельных процесса: по одному для каждого этажа и один для крыши. В каждом процессе для этажей будет не более T вызовов функции `press_buttons`, а в процессе для крыши — ровно T вызовов функции `answer`. Поэтому ваша программа не должна предполагать наличие общего состояния (например, глобальных переменных). Каждый процесс для этажей будет иметь 64 MiB памяти, отдельной от памяти других процессов, и все эти вызовы (до $(N + 1) \times T$ вызовов `press_buttons` и ровно T вызовов `answer`) должны завершиться за 10 секунд.

Constraints

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

Example

В примере имеется 1 тестовый случай со следующими значениями этажей:

```
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
```

Вот пример взаимодействия:

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>	

В этом взаимодействии корректно восстановлены значения для этажей 0 и 2. Все остальные возвращённые значения равны -1 , поскольку Карлссон не смог их восстановить.

Subtasks

Subtask	Points	Additional constraints	Количество этажей, значения на которых необходимо восстановить для получения полного балла
0	0	Пример.	–
1	15	$0 \leq v_i \leq 1$ $v_0 = v_N = 1$ Если $v_i = 0$, то $v_{i+1} = 1$ для каждого $0 \leq i < N$	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

Scoring

Если ваша программа возвращает некорректно восстановленное значение или выполняет недопустимую операцию (например, возвращает `vector` неправильной длины) хотя бы в одном из тестовых случаев подзадачи, ваша посылка получит ноль баллов за эту подзадачу.

Для одного тестового случая количество восстановленных значений вашей посылки — это число позиций, возвращённое значение которых не равно -1 . Пусть K — минимальное количество восстановленных значений среди всех тестовых случаев подзадачи (каждая подзадача содержит только 1 тест, включающий до 10 000 тестовых случаев). Тогда баллы за каждую подзадачу вычисляются согласно таблице ниже.

Subtask	Range of K	Points
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 вызывает все функции для всех тестовых случаев в рамках одного запуска.

Формат входных данных:

- строка 1: три целых числа — количество тестовых случаев T , номер последнего этажа N и номер подзадачи S ;
- строка $1 + i$: $N + 1$ целых чисел — $v_0, v_1, \dots, v_N$, где v_f — значение, записанное на этаже f .

Формат выходных данных:

- строка i : одно целое число — количество корректно восстановленных значений v в тестовом случае i ;
- строка $T + 1$: итоговые баллы.

Для получения более подробной информации измените макрос `DETAILED` с `false` на `true` в первой строке грейдера.

Вы можете позволить грейдеру генерировать тестовые случаи (значения v_f), изменив макрос `AUTO_GENERATE` с `false` на `true` во второй строке грейдера.