

クラスルームゲーム (Classroom Game)

タシケントの著名な高校では、生徒と教師の間のゲームを開催することで学校の記念日を祝うことにした。教室には N 人の生徒が座っており、0 から $N - 1$ までの番号が付けられている。それぞれの生徒は整数の数列を書くことのできる 1 枚の紙を持っている。最初、それぞれの紙は白紙である。つまり、それぞれの紙には空の数列が書かれている。

生徒は 0 から $M - 1$ までの番号がついた M 人の教師および 1 人の校長とゲームを行う。ゲームは M ステップにわたって行われ、これにも 0 から $M - 1$ までの番号が付けられている。ステップ j においては、教師 j が教室に入り、次のイベントが行われる：

- 何人かの (0 人であることもあり得る) 生徒が手を挙げる。それぞれのステップにおいて少なくとも 1 人の生徒は手を**挙げない**こと、またそれぞれの生徒はゲームを通して**高々 1 回**までしか手を挙げないことが保証される。
- 教室に入った教師は生徒が現在持っている紙を見て、このステップにおいて手を**挙げていない**生徒が持っている紙を**書き換える**ことができる。それぞれの生徒に対して、教師はその生徒の紙に書かれている内容を、新しい (空であることも許される) 整数の数列で置き換えることができる。それぞれの整数は 0 以上 63 以下でなければならない、また数列は最大でも 63 個までの要素しか持つことができない。
- これらの書き換えが終了した後、教師 j は教室を去り、生徒は秘密の順列 P に従って紙を交換する。生徒 i ($0 \leq i < N$) は自分が持っている紙を生徒 $P[i]$ に渡す。ここで、 $P[0], P[1], \dots, P[N - 1]$ は N 個の**相異なる** 0 以上 $N - 1$ 以下の整数である。 P の値はゲームのすべてのステップを通して**固定されている**。ただし、教師および校長は P の値を知らない。

M 個のステップすべてが終了し、 P に従った最後の紙の交換が終わった後、校長が部屋に入る。校長は生徒が持っている紙のみを見ることで、それぞれの生徒 i ($0 \leq i < N$) について生徒 i が手を挙げたラウンドの番号 j を決定するか、あるいは生徒 i が手を挙げなかったことを結論づけなければならない。

教師は他の教師または校長と、生徒の紙の上に書かれた数列以外の手段で連絡を取ることはできない。それぞれの教師は自分がどのステップで部屋に入るのかは知っている。

あなたの課題は、それぞれの生徒について手を挙げたのかどうかといつ手を挙げたのかを正しく特定できるような教師と校長の戦略を考案し実装することである。あなたの得点は、紙に書かれた数列の長さの長さの最大値によって決まる。この最大の長さを短くすると、より高い点数あるいは同じ点数が得られる。

実装の詳細

あなたは2つの関数を実装する必要がある。1つは教師のための関数であり、もう1つは校長のための関数である。

教師のために実装する必要がある関数は以下のものである。

```
std::vector<std::vector<int>>> process_step(  
    int N, int M, int R,  
    std::vector<int> T,  
    std::vector<std::vector<int>>> A)
```

- N : 生徒の数。
- M : ステップの数であり、また教師の数でもある。
- R : 現在のステップを表す数 (0 以上 $M - 1$ 以下である)。
- T : このステップにおいて手を挙げた生徒の番号からなる、空であることもあり得る数列であり、昇順にソートされている。
- A : 生徒の紙に書かれた数列を表す長さ N の列。 $A[i]$ ($0 \leq i < N$) は、このステップの開始時点において生徒 i が持っている紙に書かれた数列を表す。
- この関数は1回のゲームにおいて、 $R = 0, 1, \dots, M - 1$ の順にちょうど M 回呼び出される。

この関数は、教師が書き換えた後の紙の状態を A と同じフォーマットで表す長さ N の列 B を返す必要がある。

- 手を挙げた生徒 i (つまり、 T に含まれるような i) について、紙を書き換えることはできないから、 $B[i] = A[i]$ である必要がある。
- $0 \leq i < N$ を満たすそれぞれの i について、 $B[i]$ の長さは 63 を超えてはならず、また $B[i]$ に含まれるそれぞれの整数は 0 以上 63 以下でなければならない。

校長のために実装する必要がある関数は以下のものである。

```
std::vector<int> determine_steps(  
    int N, int M, std::vector<std::vector<int>>> A)
```

- N, M : 先述の関数と同じである。
- A : 長さ N の列であり、 $A[i]$ は M ステップすべてが終わった後に生徒 i が持っている紙に書かれている数列である。
- この関数は1回のゲームにおいて最後の `process_step` の呼び出しの後にちょうど1回呼び出される。

この関数は長さ N の数列 D を返す必要がある。 $0 \leq i < N$ を満たすそれぞれの i について、以下の条件が満たされていなければならない。

- 生徒 i がステップ j で手を挙げた場合、 $D[i] = j$ である。

- 生徒 i がゲームを通して一度も手を上げなかった場合、 $D[i] = -1$ である。

あなたのプログラムは `process_step` の返り値以外の手段を用いて情報を保存したり `process_step` の呼び出しから別の呼び出しへ情報をやり取りしたりしてはいけない。もしあなたのプログラムがそのようなことを試みた場合、CMS におけるあなたのスコアは正しく計算されない可能性があり、またコンテストの終了後に点数が減らされる可能性がある。採点プログラムはあなたのプログラムのインスタンスを複数個同時に起動することがあることに注意せよ。つまり、単一のゲームにおける `process_step` や `determine_steps` の呼び出しは別のインスタンスで行われる可能性があり、また別のゲームにおける `process_step` や `determine_steps` の呼び出しが同一のインスタンスで行われる可能性がある。それぞれのテストケースは最大で 5 回分のゲームからなる。

制約

- $2 \leq N \leq 63$
- $1 \leq M \leq 63$
- それぞれの生徒は 1 回のゲームにおいて **高々 1 回**手を挙げる。つまり、それぞれの生徒 i について、生徒 i が手を挙げるようなステップ j は高々 1 つしか存在しない。
- それぞれのステップにおいて、少なくとも 1 人の生徒は手を**挙げない**。

小課題と得点

小課題	配点	追加の制約
1	4	$M = 1$
2	6	$N = 2$
3	9	$0 \leq i \leq N - 1$ を満たすそれぞれの i について $P[i] = i$ である。
4	25	それぞれのステップにおいて、高々 1 人の生徒しか手を挙げない。
5	56	追加の制約はない。

それぞれのテストケースにおいて、ある `process_step` の返り値が不正であったり、あるいは `determine_step` の返り値が正しくなかったりした場合、あなたの得点は 0 である（CMS においては `Output isn't correct` と表示される）。

そうでない場合、`process_step` の**任意**の呼び出しにおける返り値 B の**任意**の数列にわたる長さの最大値を C とする。このとき、このテストケースにおける配点 S の小課題の得点は $S \cdot X$ として計算される。ここで、 X は次の表の通りに C によって決まる値である。

条件	X
$C \leq 2$	1.00
$C = 3$	0.75
$C = 4$	0.55
$5 \leq C \leq 13$	$0.50 - 0.03 \cdot (C - 5)$
$14 \leq C \leq 63$	$0.19 \cdot \frac{64-C}{64-14} + 0.04$

例

$N = 4$ 人の生徒と $M = 2$ 人の教師があり、そして順列が $P = [0, 3, 1, 2]$ であるシナリオを考える。最初の時点では全ての紙には何も書かれていない。

採点プログラムは最初に次の呼び出しを行う：

```
process_step(4, 2, 0, [1], [[], [], [], []])
```

生徒 1 は手を挙げているため、この生徒の紙は書き換えることができない。教師 0 は生徒 0 の紙に $[10]$ を書き、生徒 3 の紙に $[1, 63, 4]$ を書き、生徒 2 の紙は空白のままにすることができる。そうするためには、この関数は $B = [[10], [], [], [1, 63, 4]]$ を返さなければならない。

教師 0 が教室を去った後、生徒は P に従って紙を交換する。その結果、生徒の紙は $A = [[10], [], [1, 63, 4], []]$ で表される状態になる。

次に採点プログラムは以下の呼び出しを行う：

```
process_step(4, 2, 1, [0, 3], [[10], [], [1, 63, 4], []])
```

生徒 0 と 3 は手を挙げているため、これらの生徒の紙は書き換えることができない。教師 1 は生徒 1 の紙に $[0, 40]$ を書き、生徒 2 の紙に $[1, 50]$ を書き込むことができる。そうするためには、この関数は $B = [[10], [0, 40], [1, 50], []]$ を返さなければならない。

教師 1 が教室を去った後、生徒は P に従って紙を交換する。その結果、生徒の紙は $A = [[10], [1, 50], [], [0, 40]]$ で表される状態になる。

最後に、採点プログラムは次の呼び出しを行う：

```
determine_steps(4, 2, [[10], [1, 50], [], [0, 40]])
```

生徒 0 と 3 はステップ 1 に手を挙げ、生徒 1 はステップ 0 に手を挙げ、生徒 2 は手を挙げなかったから、この関数は $D = [1, 0, -1, 1]$ を返さなければならない。この例においては $C = 3$ である。

採点プログラムのサンプル

入力形式

```
N M
Q[0] Q[1] ... Q[N-1]
P[0] P[1] ... P[N-1]
```

ここで、 Q は長さ N の配列であり、もし生徒 i がステップ j に手を挙げるならば $Q[i] = j$ であり、手を一度も挙げないならば $Q[i] = -1$ である。

出力形式

それぞれの `process_step` の呼び出しの後に、採点プログラムのサンプルは紙に書かれている数列を出力する。 B の長さを K とし、それぞれの $0 \leq i < K$ について $L[i]$ を $B[i]$ の長さとする。採点プログラムのサンプルは B を以下の形式で出力し、**その後に空行を出力する**：

```
K
L[0] B[0][0] B[0][1] ... B[0][L[0]-1]
L[1] B[1][0] B[1][1] ... B[1][L[1]-1]
:
L[K-1] B[K-1][0] B[K-1][1] ... B[K-1][L[K-1]-1]
```

出力の後、採点プログラムのサンプルは順列 P に従って紙の交換を行う。もし $K \neq N$ であった場合、採点プログラムのサンプルは代わりにエラーメッセージを出力し実行を停止する。

`determine_steps` を呼んだ後、採点プログラムのサンプルは以下の形式で出力する：

```
C H
D[0] D[1] ... D[H-1]
```

ここで、 H は `determine_steps` によって返された数列 D の長さである。