四角形の着色(squares)

PeterとMikeはゲームを行う. n 個の四角形が一列に並んでおり, それらには左から右に 0 から $n-1$ の番号が付けられている. ゲームの開始時には, Peterはそれぞれの四角形を **黒色** または **白色** で着色する. そして彼はMikeに一つの正の整数 k $(1 \leq k \leq n)$ を指定する.

このゲームは q ラウンドに渡って行われる. それぞれのラウンドでは, Mikeが x $(0 \leq x < n)$ をランダムに決め, $x, x+1, \ldots x+k-1$ 番目の四角形の色をPeterに伝える. $x+i$ $(0 \leq i < k)$ 番目の四角形がない場合は, Mikeはそのことを Peterに伝える. そしてPeterはMikeから与えられた情報のみに基づいて x を推定する必要がある.

PeterはMikeに楽をしてほしいので, 可能な限り小さい k の値を選びたい. すべてのラウンドで正しい推定ができるものの中で, なるべく k が小さくなるようなPeterの戦略を実装せよ.

実装の詳細

あなたは以下のプロシージャを実装する必要がある.

```
int[] paint(int n)
```

- n: 四角形の個数
- このプロシージャは長さ $n+1$ の配列を戻り値として返す必要がある. 配列の初めの n 個の要素は n 個の四角形の色を表す. 配列の i 番目の要素は, i 番目の四角形の色が白色であるとき 1 , 黒色であるとき 0 となる. 配列の最後の要素は k の値を表す.
- このプロシージャはそれぞれのシナリオでちょうど一回呼び出される. この呼び出しはすべての `find_location` の呼び出しの前に行われる.

```
int find_location(int n, int c[])
```

- n: 四角形の個数
- c: 長さ k の配列. 配列の i 番目の要素は $i+x$ 番目の四角形の色を表し, その四角形の色は 1 であるとき白色, 0 であるとき黒色となる. もし $i+x$ 番目の四角形が存在しないときは, 配列の i 番目の要素は -1 となる.
- このプロシージャは推定された x の値を返す必要がある.
- このプロシージャははそれぞれのシナリオでそれぞれのラウンドごとにちょうど一回呼び出され, 全体でちょうど q 回呼び出される.

それぞれのテストケースは複数の独立したシナリオを含んでいるかもしれない(すなわち n の値が異なるような場合である). r 個のシナリオを含むテストケースの場合, 上記のプロシージャを呼び出す採点プログラム

は以下に示すようにちょうど2回実行される.

1回目の採点プログラムの実行では:

- `paint` プロシージャが r 回呼び出される.
- 戻り値の, 四角形の色の情報と k の値は採点システムに保存される.
- `find_location` プロシージャは呼び出されない.

2回目の採点プログラムの実行では:

- `find_location` プロシージャは複数回呼び出される可能性がある.
- それぞれの`find_location` プロシージャを呼ぶのに使われる n の値と与えられる色の情報は, 1回目の採点プログラムでの `paint` プロシージャの結果に応じて, 恣意的に選ばれる.
- `paint` プロシージャは呼び出されない.

入出力例

以下のような呼び出しを考える.

```
paint(5)
```

5個の四角形が並んでいる. Peterが i $(0 \leq i < 5)$ 番目の四角形をそれぞれ黒色, 黒色. 白色, 黒色, 白色と着色し, $k = 3$ がすべての x の推定をするのに必要な最小の k の値だとしたとする. このとき, プロシージャは $[0, 0, 1, 0, 1, 3]$ を返すべきである.

この後, 何回かの `find_location` プロシージャの呼び出しが行われる可能性がある.

考えられる呼び出しの一つは

```
find_location(5, [0, 1, 0])
```

である.

この呼び出しは x, $x + i$, $x + 2$ 番目の四角形の色がそれぞれ黒色, 白色, 黒色であることを表している. Peterはこの情報から $x = 1$ であると推定できるので, このプロシージャは 1 を返すべきである.

考えられる他の呼び出しは

```
find_location(5, [1, 0, 1])
```

である.

この呼び出しは x, $x + i$, $x + 2$ 番目の四角形の色がそれぞれ白色, 黒色, 白色であることを表している. Peterはこの情報から $x = 2$ であると推定できるので, このプロシージャは 2 を返すべきである.

制約

- $1 \leq r \leq 10$
- $2 \leq n, q \leq 1000$
- $-1 \leq c[i] \leq 1$ $(0 \leq i < k)$

小課題

1. (10 点) `paint` プロシージャが返す k の値が 1000 以下でなければならない.
2. (15 点) `paint` プロシージャが返す k の値が 100 以下でなければならない.
3. (20 点) `paint` プロシージャが返す k の値が 70 以下でなければならない.
4. (55 点) `paint` プロシージャが返す k の値が 30 以下でなければならない.

小課題4では部分点がつく. m をすべてのシナリオを通じて `paint` プロシージャが返す最大の k の値とする. このとき小課題4でのあなたの得点は以下の表にしたがって計算される.

k の最大値	得点
$m \geq 30$	0
$10 < m < 30$	$\frac{30-m}{20} \cdot 55$
$m \leq 10$	55

採点プログラムのサンプル

採点プログラムのサンプルは, 以下の形式で入力を読み込む:

- 1 行目: r

r 個のブロックが以下に続き, それぞれのブロックは一つのシナリオを表す. それぞれのブロックの形式は以下のとおりである:

- 1 行目: n q
- $2 + i$ $(0 \leq i \leq q - 1)$ 行目: i 番目の `find_location` プロシージャ呼び出し時の x の値.

採点プログラムのサンプルは, 以下の形式であなたの答えを出力する:

- 1 行目: m

合計 r 個のブロックがシナリオの入力の直後にそれぞれ出力される. それぞれのブロックの形式は以下のとおりである:

- $1 + i$ $(0 \leq i \leq q - 1)$ 行目: i 回目の `find_location` プロシージャの呼び出しによって得られた, 推定された x の値.

注意として，一回の採点プログラムのサンプルの実行の中では `paint` プロシージャと `find_location` プロシージャの両方の呼び出しが行われる．