ELEPHANTS

バージョン: 1.2

踊る象 (Dancing Elephants)

N 頭の象が「ステージ」と呼ばれる直線上で踊る「踊る象 (Dancing Elephants) 」はパタヤのすごいショーとして知られている.

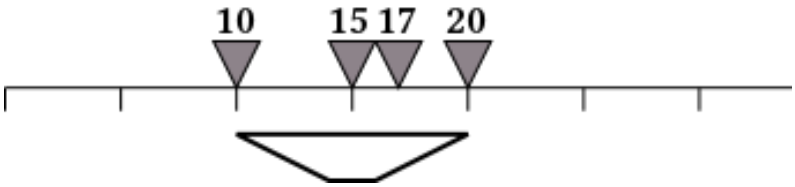
何年もの訓練により, このショーの象は多くの驚くべきダンスを踊れるようになっている. ショーはいくつかの演技からなる. それぞれの演技は, ちょうど 1 頭の象がかわいいダンスを踊る. 踊りながら, 他の位置に移動することもある.

ショーのプロデューサー達はショー全体の写真が掲載されている本を制作したい. 演技が終わるごとに, 観客から見えるすべての象の写真を撮りたい.

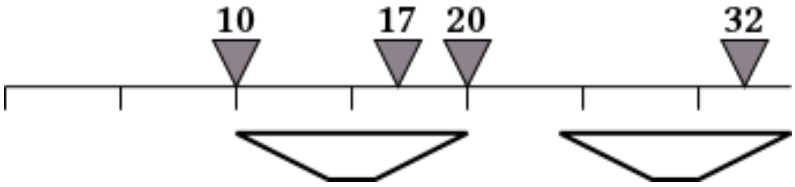
ショーの間, 複数の象が同じ位置にいることがある. この場合, 象は同じ位置の他の象の後ろに立つ.

1 つのカメラで長さ **L** の区間 (端を含む) にいる象の写真を撮ることができる. 象はステージ全体に広がることもあるので, すべての象のスナップ写真を同時に撮るには複数のカメラが必要かもしれない.

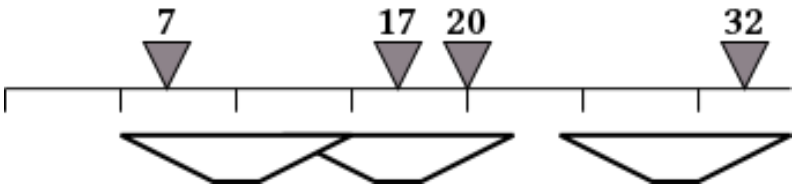
例として, **L = 10** とし象がステージ上の 10, 15, 17, 20 の位置にいる場合を考える. このとき, 下図で示すように 1 つのカメラで象の写真を撮ることができる (三角形は象を, 台形はカメラを表す).



次の演技で, 位置 15 の象は位置 32 へ踊って移動する. この演技の後, スナップ写真を撮るためには少なくとも 2 台のカメラが必要である.



次の演技で, 位置 10 の象は位置 7 へ移動する. この象の並び方では, すべての象の写真を撮るために 3 台のカメラが必要である.



この対話型課題では, それぞれの演技の後に写真を撮るために必要なカメラの**最小台数**を決定しなければなら

ない. 演技の後, 必要なカメラの台数は増加するかもしれないし, 減少するかもしれないし, そのままかもしれない.

課題 (Your task)

次のプロシージャーを実装せよ:

- 次のパラメータを持つプロシージャー **init(N, L, X)** :
  - **N** — 象の数. 象には **0** から **N−1** までの番号がふられている.
  - **L** — 1 台のカメラで撮影可能な区間の長さ. **L** は **0 ≦ L ≦ 1 000 000 000** をみたす整数であると仮定して良い.
  - **X** — 象の初期位置を表現する整数の 1 次元配列. **0 ≦ i < N** に対し, 象 **i** の初期位置は **X[i]** である. 初期位置はソートされている. より正確には **0 ≦ X[0] ≦ ... ≦ X[N−1] ≦ 1 000 000 000** と仮定して良い. 象が踊っている間に象の順番は変化するかもしれないことに注意せよ.

  このプロシージャーは最初の **update** が呼び出される前に 1 度だけ呼び出される. このプロシージャーは値を返さない.

- 次のパラメータを持つプロシージャー **update(i, y)** :
  - **i** — この演技で移動する象の番号.
  - **y** — 象 **i** がこの演技の後に立っている位置. **y** は **0 ≦ y ≦ 1 000 000 000** を満たす整数であると仮定して良い.

  このプロシージャーは複数回呼び出される. 1 回の呼び出しは 1 つの演技に対応する (この演技は, それ以前のすべての演技に続くものである). それぞれの呼び出しは対応する演技の後にすべての象を写真に撮るために必要なカメラの**最小台数**を返す必要がある.

例 (Example)

N = 4, **L = 10**, 象の初期位置が

$$X = \begin{matrix} 10 \\ 15 \\ 17 \\ 20 \end{matrix}$$

の場合を考える.

最初, プロシージャー **init** は上記のパラメータで呼び出される. その後, プロシージャー **update** が 1 つの演技ごとに 1 回ずつ呼び出される. 以下に呼び出しと正しい返り値の例を示す:

演技	パラメータ	返り値
1	update(2, 16)	1
2	update(1, 25)	2
3	update(3, 35)	2
4	update(0, 38)	2
5	update(2, 0)	3

小課題 (Subtasks)

小課題 1 (10 点)

- 象の数はちょうど **N = 2** である.
- 初期状態においてもそれぞれの演技の後においても, すべての象の位置は異なる.
- プロシージャー **update** は高々 **100** 回呼ばれる.

小課題 2 (16 点)

- **1 ≦ N ≦ 100**
- 初期状態においてもそれぞれの演技の後においても, すべての象の位置は異なる.
- プロシージャー **update** は高々 **100** 回呼ばれる.

小課題 3 (24 点)

- **1 ≦ N ≦ 50 000**
- 初期状態においてもそれぞれの演技の後においても, すべての象の位置は異なる.
- プロシージャー **update** は高々 **50 000** 回呼ばれる.

小課題 4 (47 点)

- **1 ≦ N ≦ 70 000**
- 複数の象が同じ位置を共有することがあるかもしれない.
- プロシージャー **update** は高々 **70 000** 回呼ばれる.

小課題 5 (3 点)

- **1 ≦ N ≦ 150 000**
- 複数の象が同じ位置を共有することがあるかもしれない.
- プロシージャー **update** は高々 **150 000** 回呼ばれる.
- 実装の詳細 (Implementation details) のCPU 時間制限 (CPU time limit) の注意を見よ.

実装の詳細 (Implementation details)

制限 (Limits)

- CPU 時間制限 (CPU time limit): 9 秒
  注意: C++ Standard Library (STL) のコレクションテンプレートは遅い. STLを使った場合, 特に小課題 5 を解くことができないかもしれない.
- メモリ制限 (Memory limit): 256 MB
  注意: スタックのサイズには決められた制限はない. スタックとして使用されたメモリは，メモリ総使用量に含まれる.

インターフェース (API) (Interface (API))

- 実装フォルダ (Implementation folder): `elephants/`
- 選手が実装するファイル: `elephants.c` または `elephants.cpp` または `elephants.pas`
- 提出ファイルのインターフェース (Contestant interface): `elephants.h` または `elephants.pas`

- 採点プログラムのサンプル (Sample grader): `grader.c` または `grader.cpp` または `grader.pas`
- 採点プログラムの入力のサンプル (Sample grader input): `grader.in.1`, `grader.in.2`, ...
  注意: 採点プログラムのサンプルは次の書式の入力を読み込む.
  - 1 行目: **N** と **L** と **M** を読み込む. **M** はショーで行われる演技数である.
  - 2 行目から **N+1** 行目: 初期位置. つまり, **0 ≦ k < N** に対し, **k+2** 行目には **X[k]** が書かれている.
  - **N+2** 行目から **N+M+1** 行目: **M** 回の演技の情報. つまり, **1 ≦ j ≦ M** に対し, **N+1+j** 行目には **i[j]** と **y[j]**と**s[j]** が空白区切りで書かれ, **j** 番目の演技で象 **i[j]** が位置 **y[j]** に移動し, その演技の後に必要とされるカメラの最小台数は **s[j]** であることを示している.
- 採点プログラムの入力のサンプルに対して, 期待される出力: `grader.expect.1`, `grader.expect.2`, ...

この課題において, これらのファイルはいずれも文字列 **“Correct.”** のみを含むファイルである.