코끼리 (Dancing Elephants)

N 마리의 코끼리가 무대에서 한 줄로 서서 춤을 추는 **코끼리 쇼**는 파타야에서는 매우 유명하다.

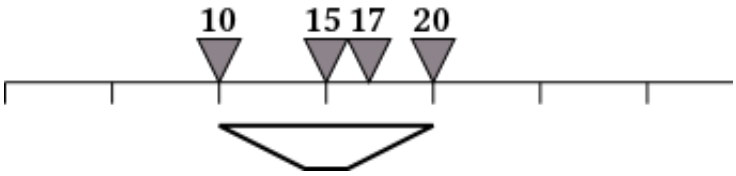
몇 년간의 훈련 후, 코끼리들은 이 쇼에서 많은 놀랄만한 춤들을 출 수 있게 된다. 이 쇼는 일련의 동작으로 이루어진다. 각 동작에서는, 정확히 한 코끼리만 다른 장소로 이동하는 귀여운 춤을 추는데, 제자리에 있을 수도 있다.

쇼 제작자는 전체 쇼의 사진을 포함하는 사진첩을 제작하려 한다. 각 동작마다, 관객들에게 보여지는 모든 코끼리들의 사진을 찍으려고 한다.

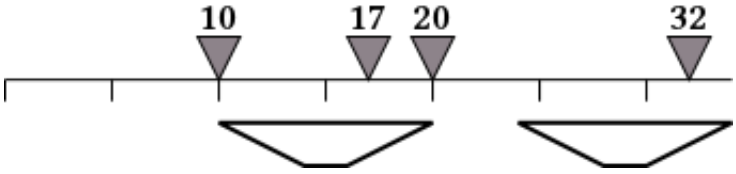
쇼가 진행되는 동안, 여러 마리의 코끼리들은 같은 장소에 있을 수 있다. 이 경우, 같은 장소에 있는 코끼리들은 단순히 다른 코끼리의 뒤에 서 있게 된다.

하나의 카메라는 길이 **L** 의 선분상에 있는 코끼리들의 사진만 찍을 수 있다(양 끝점 포함). 코끼리들은 무대 전체에 흩어져 있을 수 있으므로, 모든 코끼리들에 대한 동시의 스냅사진을 찍기 위해서는 여러 대의 카메라가 필요할 수도 있다.

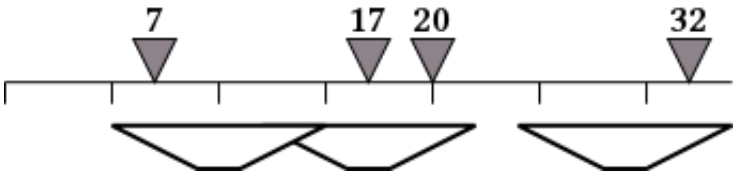
예를 들어, **L=10** 이고 무대에서 코끼리들의 위치가 10, 15, 17, 그리고 20 이라고 하자. 이 순간에는, 아래의 그림과 같이, 하나의 카메라로 모든 코끼리들의 사진을 찍을 수 있다. (삼각형은 코끼리를 나타내고; 사다리꼴은 카메라를 나타낸다.)



그 다음의 동작에서는, 15 에 있던 코끼리가 32 의 위치로 춤추며 이동한다. 이 동작 후의 스냅사진을 찍기 위해서는 적어도 두 대의 카메라가 필요하다.



그 다음의 동작에서 10 에 있던 코끼리가 7 의 위치로 이동한다. 이 경우에서는 모든 코끼리들의 사진을 찍기 위해서 세대의 카메라가 필요하다.



이 상호작용 태스크에서는, 각 코끼리들의 동작 후의 코끼리 사진을 찍기 위한 최소 수의 카메라를 찾아야 한다. 매 동작마다 카메라의 수는 증가할 수도 있고, 감소할 수도 있으며, 변화가 없을 수도 있음에 주의하라.

해야 할 일(Your task)

다음의 함수를 작성하라:

- 함수 init(N,L,X)은 다음의 파라미터를 갖는다:
  - **N** – 코끼리의 수. 코끼리는 **0** 이상 **N-1** 이하의 번호로 나타낸다.
  - **L** – 하나의 카메라로 찍을 수 있는 선분의 길이. **L** 은 정수이며, $0 \le L \le 1\,000\,000\,000$ 이다.
  - **X** – 코끼리들의 처음 위치를 나타내는 하나의 일차원 정수 배열. $0 \le i < N$ 에 대하여, 코끼리 **i** 의 초기위치는 **X[i]**이고, 정렬이 되어있다. 좀 더 정확하게 말하자면, $0 \le X[0] \le ... \le X[N-1] \le 1\,000\,000\,000$ 이다. 코끼리들의 춤(동작) 후에 위치가 재 배열 될 수 있음에 주의하라.

  이 함수는 모든 다른 호출 전에 단 한번만 호출되며 어떤 값도 리턴하지 않는다.

- 함수 update(i,y)는 다음의 파라미터를 갖는다:
  - **i** – 현재의 동작에서 움직이는 코끼리의 번호.
  - **y** – 현재의 동작 후 변경되는 코끼리 **i** 의 위치. **y** 는 정수이며, $0 \le y \le 1\,000\,000\,000$ 이다.

  이 함수는 여러 번 호출될 수 있다. 각 호출은 (이전의 모든 동작 다음에 나타나는) 한번의 동작을 의미한다. 각 호출은 이 동작 후의 모든 코끼리들의 사진을 찍는데 필요한 카메라의 최소 수를 리턴하여야 한다.

예제 (Example)

N=4, **L=10** 이고, 초기 코끼리의 위치가 다음과 같은 경우를 고려해보자.

X=
10
15
17
20

먼저, 함수 init 은 위의 파라미터로 호출될 것이다. 그 후, 함수 update 는 각 동작마다 한번씩 호출될 것이다. 다음에 함수 호출의 예와 정확한 리턴 값들이 있다:

동작	호출 파라미터	리턴 값
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

서브 태스크 (Subtasks)

서브태스크 1 (10 점)

- 정확히 **N** = 2 코끼리만 있다.
- 처음과 그 이후의 각 동작 후, 모든 코끼리들의 위치는 서로 다르다.
- 함수 update 는 최대 100 번 호출된다.

서브태스크 2 (16 점)

- $1 \leq \mathbf{N} \leq 100$.
- 처음과 그 이후의 각 동작 후, 모든 코끼리들의 위치는 서로 다르다.
- 함수 update 는 최대 100 번 호출된다.

서브태스크 3 (24 점)

- $1 \leq \mathbf{N} \leq 50\,000$.
- 처음과 그 이후의 각 동작 후, 모든 코끼리들의 위치는 서로 다르다.
- 함수 update 는 최대 50 000 번 호출된다.

서브태스크 4 (47 점)

- $1 \leq \mathbf{N} \leq 70\,000$.
- 여러 마리의 코끼리들이 같은 위치에 있을 수 있다.
- 함수 update 는 최대 70 000 번 호출된다.

서브태스크 5 (3 점)

- $1 \leq \mathbf{N} \leq 150\,000$.
- 여러 마리의 코끼리들이 같은 위치에 있을 수 있다..
- 함수 update 는 최대 150 000 번 호출된다.
- CPU 시간 제한에 대한 것은 아래의 구현시 유의사항을 참조하라.

구현시 유의사항 (Implementation details)

제약조건

- CPU 제한 시간: 9 초
  유의사항: C++ 표준 라이브러리의 탬플릿 (STL)들은 느릴 수 있기 때문에, 이것을 사용한다면, 특히, 서브태스크 5 를 풀지 못할 수 있다.
- 메모리 제한: 256 MB
  유의사항: 스택 메모리 크기에 대한 제한은 명시되지 않는다. 스택 메모리는 전체메모리 사용량에 포함된다.

인터페이스 (API)

- 프로그램 작업폴더: elephants/
- 참가자가 작성할 파일: elephants.c 또는 elephants.cpp 또는 elephants.pas
- 참가자 인터페이스: elephants.h 또는 elephants.pas
- 견본 채점프로그램 (sample grader): grader.c 또는 grader.cpp 또는 grader.pas
- 견본 채점프로그램 입력 (sample grader input): grader.in.1, grader.in.2, ...

  유의사항: 견본 채점프로그램은 다음과 같은 양식으로 입력을 읽는다:

  - 1 번째 줄: N, L 과 M, 단 M 은 쇼에서 동작(춤)의 수.
  - 2 번째 줄부터 **N**+1 번째 줄까지: 코끼리의 초기 위치;
    즉, k+2 번째 줄에는 X[k]가 있다 ($0 \le k < N$).
  - **N**+2 번째 줄부터 N+M+1 번째 줄까지: M 번의 동작에 대한 정보:
    즉, $1 \le j \le M$ 에 대하여, N+1+j 번째 줄에는 코끼리의 j 번째 동작을 나타내는 i[j], y[j], 그리고 s[j]가 빈칸을 사이에 두고 주어지고, 이것은 코끼리 i[j]가 동작 후에 y[j] 위치로 움직이고, 최소의 카메라의 수 s[j]가 필요함을 나타낸다.
- 견본 채점프로그램 입력에 대하여 예상되는 출력: grader.expect.1, grader.expect.2, ...

  이 태스크에 대해서, 각각의 파일에는 정확히 "**Correct.**"가 포함되어 있어야 한다.