Склад за ориз

В една страна има дълъг прав път, известен като Оризовия път. По пътя има **R** оризови полета. Всяко поле е разположено на целочислена координата между **1** и **L**, включително. Оризовите полета ще бъдат представени в ненамаляващ ред на техните координати. За $\mathbf{0} \leq \mathbf{i} < \mathbf{R}$, полето **i** е на координата **X[i]**. Може да предполагате, че $\mathbf{1} \leq \mathbf{X[0]} \leq \mathbf{...} \leq \mathbf{X[R-1]} \leq \mathbf{L}$.

Бележка: *няколко полета могат да имат една и съща координата.*

Планира се построяването на един *оризов склад* като общо място, където да се съхранява колкото се може повече от реколтата. Както и полетата, *складът трябва да бъде на целочислена координата* между **1** и **L**, включително. Складът може да бъде на всяко място, включително и там, където вече има едно или повече оризови полета.

Всяко оризово поле произвежда *точно по* 1 *камион* ориз всеки сезон. За превозването на товара се плаща по 1 бат на километър. С други думи транспортните разходи за превозването на реколтата от дадено поле до склада са равни на разликата в координатите.

През тази година разходите за транспорт са ограничени: може да използваме най-много **B** бата.

Задачата е така да се избере мястото на склада, че в него да се събере колкото е възможно повече ориз.

Задача

Напишете функция **`besthub(R,L,X,B)`**, която има следните параметри:

- **R** – брой на оризовите полета. Полетата са номерирани от **0** до **R – 1**.
- **L** – максималната координата.
- **X** – едномерен целочислен масив, сортиран в растящ ред. За всяко $\mathbf{0} \leq \mathbf{i} < \mathbf{R}$, полето **i** има координата **X[i]**.
- **B** – наличния бюджет.

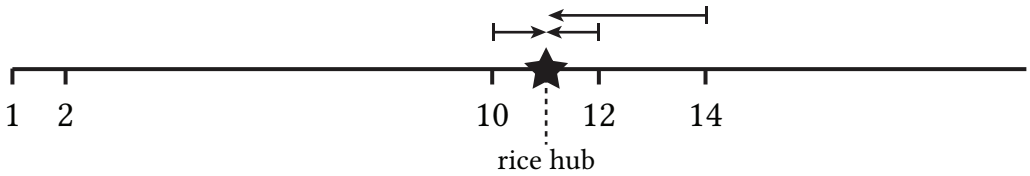
Вашата функция трябва да намира оптимално разположение на склада и трябва да върне максималния брой камиони с ориз, които могат да бъдат транспортирани до склада в рамките на дадения бюджет.

Бележка: Транспортните разходи могат да бъдат много големи. Бюджетът е даден като 64-битово цяло число. Препоръчваме да използвате 64-битови цели числа при пресмятанията. Ако програмирате на C/C++, използвайте тип `long long`.

Пример

Да разгледаме следните данни: **R=5, L=20, B=6** и

X=
1
2
10
12
14



В този случай има няколко оптимални решения: складът може да бъде на всяка координата от 10 до 14, включително. Фигурата показва един от оптималните варианти. Зърното от полетата на координати 10, 12 и 14 ще бъде превозено до склада за не повече от 6 бата. Не съществува място на склада, така че да бъде възможно събирането на ориза от повече от 3 полета в рамките на дадения бюджет, следователно решението е оптимално и **`besthub`** трябва да върне **3**.

Подзадачи

Подзадача 1 (17 точки)

- $1 \leq \mathbf{R} \leq 100$
- $1 \leq \mathbf{L} \leq 100$
- $0 \leq \mathbf{B} \leq 10\,000$
- Няма две полета с една и съща координата (*само за тази подзадача*).

Подзадача 2 (25 точки)

- $1 \leq \mathbf{R} \leq 500$
- $1 \leq \mathbf{L} \leq 10\,000$
- $0 \leq \mathbf{B} \leq 1\,000\,000$

Подзадача 3 (26 точки)

- $1 \leq \mathbf{R} \leq 5\,000$
- $1 \leq \mathbf{L} \leq 1\,000\,000$
- $0 \leq \mathbf{B} \leq 2\,000\,000\,000$

Подзадача 4 (32 точки)

- $1 \leq \mathbf{R} \leq 100\,000$
- $1 \leq \mathbf{L} \leq 1\,000\,000\,000$
- $0 \leq \mathbf{B} \leq 2\,000\,000\,000\,000\,000$

Детайли по реализацията

Ограничения

- Ограничение по време: 1 секунда
- Ограничение по памет: 256 MB
  Бележка: Няма явна граница за размера на стековата памет. Паметта за стека се прибавя към общата използвана памет.

Интерфейс (API)

- Папка: ricehub/
- Да бъдат написани от състезателя: ricehub.c или ricehub.cpp или ricehub.pas
- Интерфейс на състезателя: ricehub.h или ricehub.pas
- Примерен грейдър: grader.c или grader.cpp или grader.pas
- Вход за примерния грейдър: grader.in.1, grader.in.2, ...
- **Бележка:** Примерният грейдър чете входа в следния формат:
  - Ред 1: **R**, **L** и **B**.
  - Редове от 2 до **R** + 1: местата на оризовите полета;

    ред **i** + 2 съдържа **X**[**i**], за $0 \leq \mathbf{i} < \mathbf{R}$.
  - Ред **R** + 2: очакваното решение.
- Очакван изход за входа на примерния грейдър : grader.expect.1, grader.expect.2, ...

  За тази задача, всеки един от тези файлове трябва да съдържа единствено текста “**Correct.**”.