Rice Hub

In the countryside, you can find a long straight road known as the Rice Way. Along this road there are **R** rice fields. Each field is located at an integer coordinate between **1** and **L**, inclusive. The rice fields will be presented in non-decreasing order of their coordinates. Formally, for **0** ≤ **i** < **R**, rice field **i** is at coordinate **X[i]**. You may assume that **1** ≤ **X[0]** ≤ **...** ≤ **X[R-1]** ≤ **L**.

Please note that *multiple rice fields may share the same coordinate.*

We plan to construct a single *rice hub* as a common place to store as much of the harvest as possible. As with the rice fields, *the hub has to be at an integer coordinate* between **1** and **L**, inclusive. The rice hub can be at any location, including one that already contains one or more rice fields.

Each rice field produces *exactly 1 truckload* of rice every harvest season. To transport the rice to the hub, the city has to hire a truck driver. The driver charges 1 Baht to transport a truckload of rice per unit of distance towards the hub. In other words, the cost of transporting rice from a given field to the rice hub is numerically equal to the difference between their coordinates.

Unfortunately, our budget for this season is tight: we may only spend at most **B** Baht on transportation. Your task is to help us strategically place the hub to gather as much rice as possible.

Your task

Write a procedure **besthub(R,L,X,B)** that takes the following parameters:

- **R** – the number of rice fields. The fields are numbered **0** through **R-1**.
- **L** – the maximum coordinate.
- **X** – a one-dimensional array of integers sorted from smallest to largest.
  For each **0** ≤ **i** < **R**, field **i** is located at **X[i]**.
- **B** – the budget.

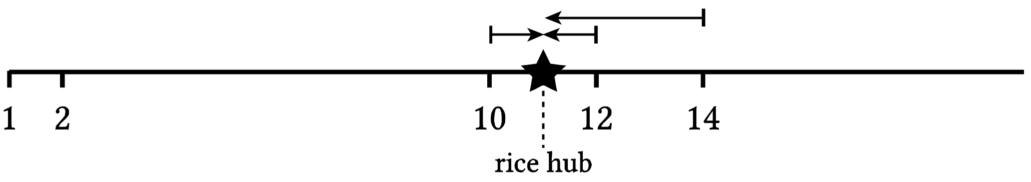
Your procedure must find an optimal location of the hub and return the maximum number of truckloads of rice that can be transported to the hub within the budget.

Note that the total cost of transporting the rice can be very large. The budget is given as a 64-bit integer, and we recommend that you use 64-bit integers in your computation. In C/C++, use the type long long; in Pascal, use the type Int64.

Example

Consider the case where **R**=**5**, **L**=**20**, **B**=**6**, and

```
   1
   2
X= 10
   12
   14
```



For this case, there are multiple optimal locations for the hub: you can place it anywhere between locations 10 and 14, inclusive. The figure above shows one of these optimal locations. You will then be able to transport rice from fields at coordinates 10, 12, and 14. For any optimal hub location, the total cost of this transportation will be at most 6 Baht. Clearly, no hub location will allow us to gather rice from more than three fields, hence this solution is optimal and **besthub** should return **3**.

Subtasks

Subtask 1 (17 points)

- $1 \le \mathbf{R} \le 100$
- $1 \le \mathbf{L} \le 100$
- $0 \le \mathbf{B} \le 10\,000$
- No two rice fields share the same coordinate (*only for this subtask*).

Subtask 2 (25 points)

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

Subtask 3 (26 points)

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

Subtask 4 (32 points)

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

Implementation details

Limits

- CPU time limit: 1 second
- Memory limit: 256 MB
  Note: There is no explicit limit for the size of stack memory. Stack memory counts towards the total memory usage.

Interface (API)

- Implementation folder: ricehub/
- To be implemented by contestant: ricehub.c or ricehub.cpp or ricehub.pas
- Contestant interface: ricehub.h or ricehub.pas
- Sample grader: grader.c or grader.cpp or grader.pas
- Sample grader input: grader.in.1, grader.in.2, ...

Note: The sample grader reads the input in the following format:

- Line 1: **R**, **L**, and **B**.
- Lines 2 to **R**+1: locations of rice fields; i.e., line **i+2** contains **X[i]**, for **0** ≤ **i** < **R**.
- Line **R**+2: the expected solution.

- Expected output for sample grader input: grader.expect.1, grader.expect.2, ...
  For this task, each one of these files should contain precisely the text "**Correct.**"