Gap

There are N non-negative integers $a_1, a_2, \ldots, a_N$ satisfying the following inequality $0 \leq a_1 < a_2 < \cdots < a_N \leq 10^{18}$. Jeehak wants to know the *largest possible* value of $a_{i+1} - a_i$ where i ranges from 1 to $N - 1$. The input integers will not be given directly to Jeehak's program but will be accessible through a special funtion. See sections Implementation of your selected programming language for details.

Task

Help Jeehak to implement a function to return the largest possible value of $a_{i+1} - a_i$ where i ranges from 1 to $N - 1$.

Implementation for C and C++

You need to implement one function `findGap(T, N)` that takes the following parameter and returns an integer of type `long long`:

- `T` — the subtask number (1 or 2)
- `N` — the number of given integers

Your function `findGap` can call function `MinMax(s, t, &mn, &mx)` where the first two parameters `s` and `t` are integers of type `long long` and the last two paramters `&mn` and `&mx` are pointers to integer variables of type `long long`, i.e., `mn` and `mx` are integer variables of type `long long`. When `MinMax(s, t, &mn, &mx)` returns, the variable `mn` will have the value of smallest a_i larger than or equal to the value of `s` and the variable `mx` will have the value of largest a_j smaller than or equal to the value of `t`. In case there are no input integers between `s` and `t` (inclusive), then both `mn` and `mx` will have the value `-1`. The value of `s` should be no larger than the value of `t` when `MinMax` is called. If this condition is not met, program will be terminated with a non-zero exit code.

Implementation for Pascal

You need to implement one function `findGap(T, N)` that takes the following parameter and returns an integer of type `Int64`:

- `T` — the subtask number (1 or 2) (`Integer` type)
- `N` — the number of given integers (`LongInt` type)

Your function `findGap` can call procedure `MinMax(s, t, mn, mx)` where the first two parameters `s` and `t` are integers of type `Int64` and the last two paramters `mn` and `mx` are variables **called by reference** of type `Int64`, i.e., `mn` and `mx` are integer variables of type `Int64`. When `MinMax(s, t, mn, mx)` exits, the variable `mn` will have the value of smallest a_i larger than or equal to the value of `s` and the variable `mx` will have the value of largest a_j smaller than or equal to the value of `t`. In case there are no input integers between `s` and `t` (inclusive), then both `mn` and `mx` will have the value `-1`. The value of `s` should be no larger than the value of `t` when `MinMax` is called.

If this condition is not met, the program will be terminated.

Implementation for all

In addition to the standard requirements (time and memory limits, no runtime errors, etc), your submission has to achieve the following in order to solve a testcase:

- your function `findGap` must return the correct answer,
- the cost M associated with calls to function `MinMax` must not exceed the allowed limit (see section Scoring).

Example for C, C++

Consider the case where $N = 4$ and $a_1 = 2, a_2 = 3, a_3 = 6$, and $a_4 = 8$.

The answer, which is 3, can be calculated and thus returned by `findGap` if the following calls to `MinMax` are made:

- `MinMax(1, 2, &mn, &mx)` is called and `mn` and `mx` both have the value 2.
- `MinMax(3, 7, &mn, &mx)` is called and `mn` have the value 3 and `mx` has the value 6.
- `MinMax(8, 9, &mn, &mx)` is called and `mn` and `mx` both have the value 8.

Example for Pascal

Consider the case where $N = 4$ and $a_1 = 2, a_2 = 3, a_3 = 6$, and $a_4 = 8$.

The answer, which is 3, can be calculated and thus returned by `findGap` if the following calls to `MinMax` are made:

- `MinMax(1, 2, mn, mx)` is called and `mn` and `mx` both have the value 2.
- `MinMax(3, 7, mn, mx)` is called and `mn` have the value 3 and `mx` has the value 6.
- `MinMax(8, 9, mn, mx)` is called and `mn` and `mx` both have the value 8.

Scoring

In all subtasks the constraint $2 \leq N \leq 100,000$ holds.

Subtask 1 (30 points): Each call to `MinMax` will add 1 to M. You will receive the full score for the subtask if $M \leq \frac{N+1}{2}$ for all test cases.

Subtask 2 (70 points): Let k be the number of input integers larger than or equal to `s` and smaller than or equal to `t` in a call to `MinMax`. Each call to `MinMax` will add $k + 1$ to M. The final score will be calculated by the following rule: Final score for the subtask is the minimum score you received among all test cases. For a test case, the score is 70 if $M \leq 3N$ and the score is $\frac{60}{\sqrt{\frac{M}{N}+1}-1}$, otherwise.

Experimentation

The sample grader which can be downloaded from the scoring system will read data from standard input. The first line of input should contain two integers, subtask number T, and N. The next line should contain N integers in ascending order. The sample grader will write to standard output the value returned by `findGap` in the first line and the value of M appropriate for the subtask the input test case belongs to.

The following input describes the above example:

```
2 4
2 3 6 8
```