# Shopping

JOI Store sells $N$ items, numbered from 0 through $N - 1$. The price of the item $i$ ($0 \le i \le N - 1$) is $P_i$. The prices of any two items are different.

Anna comes to JOI Store for shopping. Among the items whose indices are between $L$ and $R$, inclusive, Anna wants to buy the cheapest one. Since Anna does not know the price of each item, she will communicate with Bruno, a clerk of JOI Store, to decide which item to buy. Bruno knows the price of every item, but he does not know the values of $L$ and $R$.

Anna and Bruno will use a telecommunication device to send characters. Every character they will send is 0 or 1. Anna can send at most 18 characters to Bruno, and Bruno can send at most 10 000 characters to Anna. Bruno wants to send as few characters as possible.

The values of $N$, $L$, and $R$ will be given to Anna, and the value of $N$ and the price of each item will be given to Bruno. Write program which implements Anna's strategy and Bruno's strategy so that Anna can decide which item to buy.

## Implementation Details

You need to submit two files.

The first file is `Anna.cpp`. It should implement Anna's strategy. It should implement the following function. The program should include `Anna.h` using the preprocessing directive `#include`.

- `void InitA(int N, int L, int R)`
  For each test case, this function is called exactly once in the beginning.
  - The parameter `N` is the number of items $N$.
  - The parameters `L` and `R` mean Anna wants to buy the cheapest item among the items whose indices are between $L$ and $R$, inclusive.
- `void ReceiveA(bool x)`
  This function is called every time when Bruno sends a character to Anna.
  - The parameter `x` denotes the character sent by Bruno to Anna. Here `true` denotes the character 1, and `false` denotes the character 0.
- `int Answer()`
  This function is called exactly once when all of the calls are finished. This function should return the index of the item Anna will buy.
  - The return value should be an integer between $L$ and $R$, inclusive. If this condition is not satisfied,

your program is judged as **Wrong Answer [1]**. If the return value is different from the item Anna will buy, your program is judged as **Wrong Answer [2]**.

In this file, your program can call the following function.

★ `void SendA(bool y)`
Anna can send a character to Bruno by this function.
- ◦ The parameter `y` denotes the character sent by Anna to Bruno. Here `true` denotes the character 1, and `false` denotes the character 0.

The second file is `Bruno.cpp`. It should implement Bruno's strategy. It should implement the following function. The program should include `Bruno.h` using the preprocessing directive `#include`.

- `void InitB(int N, std::vector<int> P)`
  For each test case, this function is called exactly once in the beginning.
  - ◦ The parameter `N` is the number of items $N$.
  - ◦ The parameter `P` is an array of length $N$. It means `P[i]` is the price $P_i$ of the item $i$ ($0 \leq i \leq N - 1$).
- `void ReceiveB(bool y)`
  This function is called every time when Anna sends a character to Bruno.
  - ◦ The parameter `y` denotes the character sent by Anna to Bruno. Here `true` denotes the character 1, and `false` denotes the character 0

In this file, your program can call the following function.

★ `void SendB(bool x)`
Bruno can send a character to Anna by this function.
- ◦ The parameter `x` denotes the character sent by Bruno to Anna. Here `true` denotes the character 1, and `false` denotes the character 0

You can assume the program will be executed as follows. For each test case, two queues are prepared: $Q_Y$ for characters sent by Anna and $Q_X$ for characters sent by Bruno. In the beginning, the functions `InitA` and `InitB` are called and characters sent by these functions are pushed to the corresponding queues. Then we will repeat the following processes.

- If either $Q_X$ or $Q_Y$ is non-empty, we pop a character from it, and call the corresponding function `ReceiveA` or `ReceiveB`. If both $Q_X$ and $Q_Y$ are non-empty, it is not determined which of `ReceiveA` or `ReceiveB` will be called.
- Whenever `SendA` is called during executions of `ReceiveA`, the sent character is pushed to $Q_Y$.

- Whenever `SendB` is called during executions of `ReceiveB`, the sent character is pushed to $Q_X$.
- If both of the queues are empty, `Answer` is called, and the program will be terminated.

Anna should not send more than 18 characters to Bruno. If it exceeds, your program is judged as **Wrong Answer [3]**. Bruno should not send more than 10 000 characters to Anna. If it exceeds, your program is judged as **Wrong Answer [4]**.

### Important Notices

- Your program can implement other functions for internal use, or use global variables. Submitted files will be compiled with the grader, and become a single executable file. All global variables and internal functions should be declared in an unnamed namespace to avoid confliction with other files. When it is graded, it will be executed as two processes of Anna and Bruno. The process of Anna and the process of Bruno cannot share global variables.
- Your program must not use the standard input and the standard output. Your program must not communicate with other files by any methods. However, your program may output debugging information to the standard error.

## Compilation and Test Run

You can download an archive file from the contest webpage which contains the sample grader to test your program. The archive file also contains a sample source file of your program.

The sample grader is the file `grader.cpp`. In order to test your program, put `grader.cpp`, `Anna.cpp`, `Bruno.cpp`, `Anna.h`, `Bruno.h` in the same directory, and run the following command to compile your programs.

```
g++ -std=gnu++17 -O2 -fsigned-char -o grader grader.cpp Anna.cpp Bruno.cpp
```

When the compilation succeeds, the executable file `grader` is generated.

Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output and the standard error.

### Input for the Sample Grader

The sample grader reads the following data from the standard input.

$N\ L\ R$
$P_0\ P_1\ \cdots\ P_{N-1}$

## Output of the Sample Grader

When the program terminates successfully, the sample grader writes the following information to the standard output (quotes for clarity).

- If the answer is correct, it writes the total number $Y$ of characters sent from Anna to Bruno and the total number $X$ of characters sent from Bruno to Anna as "`Accepted: Y X`".
- If your program is judged as Wrong Answer, it writes its type as "`Wrong Answer [1]`".

If your program is judged as several types of Wrong Answer, the sample grader reports only one of them.

## Constraints

- $1 \le N \le 1\,000\,000$.
- $0 \le L \le R \le N - 1$.
- $1 \le P_i \le N$ $(0 \le i \le N - 1)$.
- $P_i \ne P_j$ $(0 \le i < j \le N - 1)$.

## Subtasks

1. (1 point) $N \le 1\,000$.
2. (9 points) $N \le 10\,000$.
3. (90 points) No additional constraints.
    - Let $T$ be the maximum of the number of characters sent from Bruno to Anna for every test case of this Subtask.
    - Your score of this Subtask is calculated as follows.
        - If $5\,000 < T \le 10\,000$, your score is $\left\lfloor 25 \times \frac{10000 - T}{5000} \right\rfloor$ points.
        - If $1\,000 < T \le 5\,000$, your score is $25 + \left\lfloor 40 \times \frac{5000 - T}{4000} \right\rfloor$ points.
        - If $300 < T \le 1\,000$, your score is $65 + \left\lfloor 25 \times \frac{1000 - T}{700} \right\rfloor$ points.
        - If $T \le 300$, your score is 90 points.

Here $\lfloor x \rfloor$ is the largest integer not exceeding $x$.

## Sample Communication

Here is a sample input for the sample grader and corresponding function calls.

| Sample Input 1 | Sample Function Calls | | |
|---|---|---|---|
| | Call | Call | Return |
| `4 0 2`<br>`3 1 4 2` | `InitA(4, 0, 2)` | | |
| | | `SendA(true)` | |
| | | `SendA(false)` | |
| | `InitB(4, {3, 1, 4, 2})` | | |
| | `ReceiveB(true)` | | |
| | | `SendB(true)` | |
| | `ReceiveA(true)` | | |
| | `ReceiveB(false)` | | |
| | `Answer()` | | 1 |