

Make All Equal (equal)

If you visit the prehistoric stone circle of Stonehenge at dawn on the longest day of the year, the druids will challenge you to participate in the ancient and sacred *pebble game*. First, you will be blindfolded, so that you are unable to see anything.

Then, the chief druid will tell you that she has N piles of stones in a line, **where N is a power of two** ($N = 2^k$ for some integer k).

Piles are indexed from 0 to $N - 1$. Each pile has a height H_i , which is a positive integer, and the piles are ordered from smallest to tallest ($H_0 \leq H_1 \leq \dots \leq H_{N-1}$). The chief druid tells you the value of N , but not the initial heights.


You can then choose actions of one of the following types:

1. Select a positive number X and a subset of the piles S . The druids will then add X stones to each of the piles in S , and then reorder the piles from smallest to largest.
2. Choose two piles i and j . The druids will then tell you whether these two piles currently have the same height.

Your goal is to reach a configuration where all piles have the same height. You take at most Q_{add} actions of the first type, and at most Q_{compare} actions of the second type (see section Scoring).

Implementation

You will have to submit a single `.cpp` source file.

 Among this task's attachments you will find a template `equal.cpp` with a sample implementation.

You have to implement the following function:

```
C++ | void make_all_equal(int N, int Q_add, int Q_compare);
```

- Integer N represents the number of piles.
- Integer Q_{add} represents the maximum number of times you can call `add`.
- Integer Q_{compare} represents the maximum number of times you can call `compare`.

You can call the following functions:

```
C++ | void add(vector<int> S, long long X);
```

- The vector S must contain **distinct** integers between 0 and $N - 1$ inclusive.
- Integer X must be between 0 and 10^{12} inclusive.
- This function will increment H_i by X for every i in S . Then, it will sort the heights in increasing order ($H_0 \leq \dots \leq H_{N-1}$).
- This function can be called at most Q_{add} times.

```
C++ | bool compare(int i, int j);
```

- i and j must be between 0 and $N - 1$ inclusive.

- The function will return **true** if the i -th smallest pile and the j -th smallest pile have the same current height (that is, if $H_i = H_j$), and **false** otherwise.
- This function can be called at most Q_{compare} times.

Sample Grader

The task's directory contains a simplified version of the jury grader, which you can use to test your solution locally. The simplified grader reads the input data from **stdin**, calls the functions that you must implement, and finally writes the output to **stdout**.

The input is made up of two lines, containing:

- Line 1: the integers N , Q_{add} and Q_{compare} , separated by space.
- Line 2: the integers H_i , separated by space.

If your program is judged as **Accepted**, the sample grader prints **Accepted: add=U, compare=V** where U and V are the number of times you called **add** and **compare**.

If your program is judged as **Wrong Answer**, the sample grader prints **Wrong Answer: MSG**, where MSG is one of:

Message	Meaning
too many calls to add	You called add more than Q_{add} times.
X out of range	The integer X given to add is not between 0 and 10^{12} inclusive.
index in S out of range	An element of the vector S given to add is not between 0 and $N - 1$ inclusive.
indices in S not distinct	There are two equal elements in the vector S given to add .
too many calls to compare	You called compare more than Q_{compare} times.
i out of range	The integer i given to compare is not between 0 and $N - 1$ inclusive.
j out of range	The integer j given to compare is not between 0 and $N - 1$ inclusive.
heights are not equal	After calling make_all_equal , there are two piles with different heights.

Constraints

- $2 \leq N \leq 2048$, and N is a power of two.
- The initial heights satisfy $1 \leq H_0 \leq \dots \leq H_{N-1} \leq 1\,000\,000$.

Scoring

Your program will be tested on a set of test cases grouped by subtask. To obtain the score associated to a subtask, you need to correctly solve all the test cases it contains.

- **Subtask 1** [0 points]: Sample test case.
- **Subtask 2** [5 points]: $N = 2$, $H_i \leq 4$, $Q_{\text{add}} \geq 3000$ and $Q_{\text{compare}} \geq 4000$.
- **Subtask 3** [16 points]: $N = 2$, $H_i \leq 1\,000\,000$, $Q_{\text{add}} \geq 22$ and $Q_{\text{compare}} \geq 1$.
- **Subtask 4** [15 points]: $N = 256$, $H_i \leq 10$, $Q_{\text{add}} \geq 3000$ and $Q_{\text{compare}} \geq 255$.
- **Subtask 5** [18 points]: $N = 4$, $H_i \leq 1\,000\,000$, $Q_{\text{add}} \geq 45$ and $Q_{\text{compare}} \geq 3$.

- **Subtask 6** [22 points]: $N = 2048$, $H_i \leq 1\,000\,000$, $Q_{\text{add}} \geq 298$ and $Q_{\text{compare}} \geq 4000$ and initially, there are at most two distinct heights. In other words, for every i , either $H_0 = H_i$ or $H_i = H_{N-1}$.
- **Subtask 7** [24 points]: $N = 2048$, $H_i \leq 1\,000\,000$, $Q_{\text{add}} \geq 298$ and $Q_{\text{compare}} \geq 2047$.

Please note that none of these subtasks contains all testcases.

Examples

Input	Sample Communication			
	Calls	Return	Heights	Explanation
4 45 3 1 4 5 5	compare(1, 2)	false		The solution asks whether $H_1 = H_2$. Since $H_1 = 4$ and $H_2 = 5$, this is false.
	compare(2, 3)	true		The solution asks whether $H_2 = H_3$. Since $H_2 = H_3 = 5$, this is true.
	add({0, 1}, 2)		3, 5, 5, 6	After adding the stones, the new heights are [3, 6, 5, 5]. After resorting them, they become $H = [3, 5, 5, 6]$.
	compare(1, 2)	true		The solution asks whether $H_1 = H_2$. Since $H_1 = H_2 = 5$, this is true.
	add({0, 3}, 3)		5, 5, 6, 9	After adding the stones, the new heights are [6, 5, 5, 9]. After resorting them, they become $H = [5, 5, 6, 9]$.
	add({0, 1}, 4)		6, 9, 9, 9	After adding the stones, the new heights are [9, 9, 6, 9]. After resorting them, they become $H = [6, 9, 9, 9]$.
	add({0}, 3)		9, 9, 9, 9	After adding the stones, the new heights are [9, 9, 9, 9]. After resorting them, they become $H = [9, 9, 9, 9]$.