Problem Game

C header	`game.h`
C++ header	`game.h`

After one year of playing hide-and-seek, Lulu and Tanaka have finally found each other. Now, it is time for them to go back to solving impossible programming tasks.

Today, Lulu came to Tanaka with yet another interesting algorithmic game, which is played in two different rounds:

Round 1 Given two numbers N and K, Tanaka must choose a set of intervals $[x, y]$. For each such interval, $1 \leq x \leq y \leq N$ must hold. The size of the set must be at most 200000.

Round 2 Lulu gives Tanaka, one by one, several queries in the form of intervals $[a, b]$ with $1 \leq a \leq b \leq N$. For each query, Tanaka must choose a subset $[x_1, y_1], [x_2, y_2], \ldots, [x_p, y_p]$ of the intervals in Round 1, so that for all integer values *val* contained by $[a, b]$, there exists at least one interval in the subset which contains *val*, and for all integer values *val* which are not contained by $[a, b]$, there are no intervals in the subset which contain *val*. **Moreover, it must hold that $p \leq K$ for each query.** (We say an integer value *val* is contained by an interval $[x, y]$ if and only if $x \leq val \leq y$).

Since Lulu likes disjoint intervals, **he will give Tanaka 100% of the score for the game only if for each query, all intervals chosen by Tanaka are disjoint.** However, if all the answers to the queries are right, but the intervals are not disjoint, Tanaka will still get 50% of the score.

Lulu also wants the set chosen in round 1 to be relatively small. After hearing about this task, Tanaka said to Lulu: "Answering your queries won't be hard, Lulu!" However, he needs your help. Your task is to write a program which can choose the initial set and then answer to all Lulu's queries.

Interaction Protocol

The contestant must implement two functions:

```
void init(int N, int K);
void query(int a, int b);
```

The contestant may call the following function:

```
void chooseInterval(int left, int right);
```

The function `init` will be called **exactly once**, at the beginning of the interaction. The function will be supplied with the values N and K, the two numbers which were given to Tanaka in Round 1. The contestant must call the function `chooseInterval` not more than 200000 times with the intervals which Tanaka initially chose in Round 1. Then, the committee will call the function `query` multiple times. It will be supplied with the values a and b, representing a query. The contestant must call again the function `chooseInterval` for the intervals chosen by Tanaka from the list he provided in Round 1.

Attention! The contestant must not implement the main function, and must `#include` the `game.h` header! Contestants are allowed to use global variables and other functions, which will be kept between interactions.

Scoring

Let C be the amount of intervals chosen by the contestant's program and M the maximal amount of intervals which can be chosen for each subtask. Let *Points* be the maximal score for a subtask. Let *disjoint* $= 1$ if the intervals chosen by Tanaka in each query are disjoint, and *disjoint* $= \frac{1}{2}$ otherwise. If

$C > 2 \times 10^5$ then the score for the test will be 0. Otherwise, the score for each test in the subtask will be equal to:

$$S = Points \times \min\left(1, \frac{M}{C}\right) \times disjoint$$

The score for a subtask will be the minimum score S among all its tests.

Restrictions

- Let Q be the number of times the **query** function will be called by the committee.

#	Points	N	K	Q	M
1	6	50	1	2500	1275
2	6	1000	2	5000	7997
3	9	10000	2	50000	113645
4	9	1000	3	5000	5485
5	11	10000	3	50000	76989
6	8	10000	4	50000	58962
7	7	10000	5	50000	47986
8	6	10000	6	50000	40956
9	6	10000	7	50000	36011
10	6	10000	8	50000	33911
11	7	10000	9	50000	30923
12	9	10000	10	50000	26598
13	10	1000	20	50000	1786

Examples

Input	Output
`init(5,3)`	

	`chooseInterval(1,1)`
	`chooseInterval(2,3)`
	`chooseInterval(4,4)`
	`chooseInterval(4,5)`

`query(1,3)`	

	`chooseInterval(1,1)`
	`chooseInterval(2,3)`

`query(2,4)`	

	`chooseInterval(2,3)`
	`chooseInterval(4,4)`

`query(1,5)`	

	`chooseInterval(1,1)`
	`chooseInterval(2,3)`
	`chooseInterval(4,5)`

Explanations

The function `init` is called by the comittee, with values $N = 5$ and $K = 3$. The chosen intervals are: $[1, 1], [2, 3], [4, 4], [4, 5]$.

For the first query, $a = 1$ and $b = 3$. The chosen intervals are: $[1, 1], [2, 3]$. The number 1 is contained by $[1, 1]$, 2 is contained by $[2, 3]$ and 3 is contained by $[2, 3]$.

For the second query, $a = 2$ and $b = 4$. The chosen intervals are: $[2, 3], [4, 4]$. The number 2 is contained by $[2, 3]$, 3 is contained by $[2, 3]$ and 4 is contained by $[4, 4]$.

For the third query, $a = 1$ and $b = 5$. The chosen intervals are: $[1, 1], [2, 3], [4, 5]$. The number 1 is contained by $[1, 1]$, 2 is contained by $[2, 3]$, 3 is contained by $[2, 3]$, 4 is contained by $[4, 5]$ and 5 is contained by $[4, 5]$.