birthday

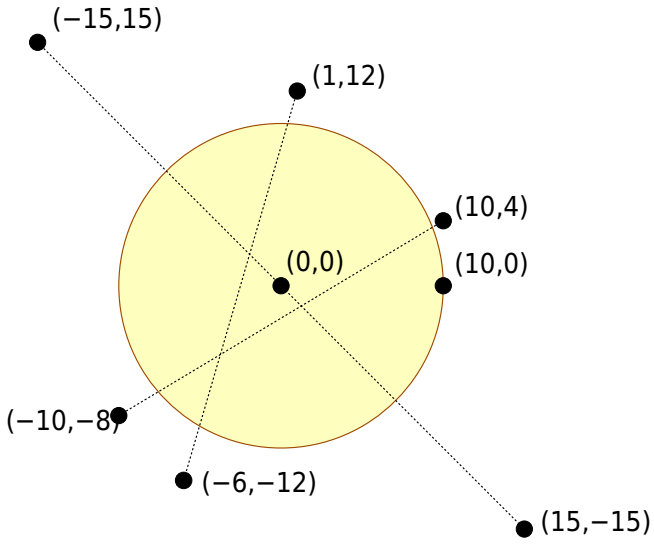
English — 1.0

It is your little sister's birthday, and by tradition she has the honour of cutting her circular cake. She is very young, however, and not yet skilled at cake cutting. Each of her cuts follows a line through the cake, but does not necessarily divide the cake into equal pieces. She gleefully cuts the cake again and again until she has created an ugly but delicious mess.

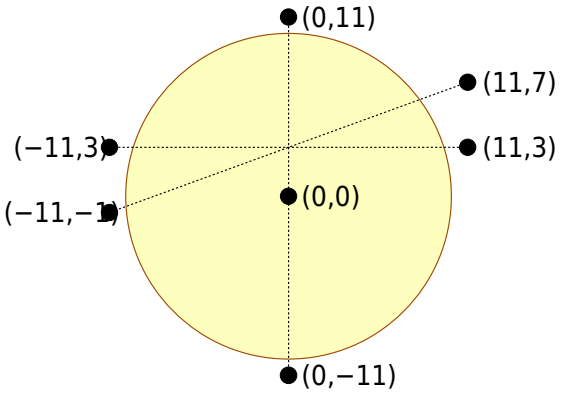
Your task is, given the cuts, to determine how many pieces the cake has been divided into. You may assume that:

- the cake is aligned on an (x,y) coordinate system, with the centre of the cake at (0,0);
- cuts have zero thickness;
- no two cuts follow the same line;
- both endpoints of every cut lie strictly outside the cake;
- every cut passes through the interior of the cake (so there is a positive amount of cake on both sides of the cut).

Examples



The picture above shows a cake of radius 10 with three cuts. The first cut runs from (-15, 15) to (15, -15), the second cut runs from (1, 12) to (-6, -12), and the third cut runs from (10, 4) to (-10, -8). Together these cuts divide the cake into seven pieces.



The second picture above shows another cake of radius 10, with three cuts that all run through the same point near the centre of the cake. This second cake is divided into only six pieces.

Implementation

You should submit a file implementing the function `pieces` as follows:

Your Function: `pieces()`

C/C++
```
int pieces(int R, int N, int *X1, int *Y1, int *X2, int *Y2);
```

Pascal
```
function pieces(R, N : LongInt; var X1, Y1, X2, Y2 : array of
LongInt) : LongInt;
```

Description

Your submission must implement this function.

This function should calculate the number the number of pieces that the cake has been divided into.

Parameters

- ***R***: The radius of the cake.
- ***N***: The number of cuts.
- ***X1***, ***Y1***, ***X2*** and ***Y2***: Arrays of length *N* containing integers that specify the endpoints of each cut, so that the *i*th cut runs from the point (*X1[i-1], Y1[i-1]*) to (*X2[i-1], Y2[i-1]*).
- **Returns**: The number of pieces the cake has been divided into.

Sample Session

The following session describes the first example above:

R	*N*	*X1*	*X2*	*Y1*	*Y2*	Return value
10	3	[-15, 1, 10]	[15, 12, 4]	[15, -6, -10]	[-15, -12, -8]	7

The following session describes the second example above:

R	*N*	*X1*	*X2*	*Y1*	*Y2*	Return value
10	3	[0, -11, -11]	[11, 3, -1]	[0, 11, 11]	[-11, 3, 7]	6

Constraints

- Time limit: 1 second
- Memory limit: 32 MiB
- 1 ≤ R ≤ 500
- -500 ≤ X1[i], Y1[i], X2[i], Y2[i] ≤ 500
- 1 ≤ N ≤ 1000

Subtasks

Subtask	Points	Additional Input Constraints
1	30	You may assume $N \leq 20$
2	30	You may assume $N \leq 250$
3	40	*(None)*

Experimentation

The sample grader provided will read input from the file `birthday.in`, which must be in the following format:

- line 1: R
- line 2: N
- line 3,. .., N+2: X1[i] Y1[i] X2[i] Y2[i]

For instance, the first example above should be provided in the following format:

```
10
3
-15 15 15 -15
1 12 -6 -12
10 4 -10 -8
```

Language Notes

C/C++	You must `#include "birthday.h"`.
Pascal	You must define the `unit Birthday`. All arrays are numbered beginning at 0 (not 1).

See the solution templates on your machine for examples.

After seeing the sights of Seoul, Paris and Rio de Janeiro, you decide it's time to get more out of your travels. You would like to gain simultaneous citizenship of as many countries as you possibly can.

Since the mathematicians took control of the United Nations, citizenship rules have become extremely simple. Although each country has its own rules on gaining citizenship, all countries obey the same basic principles:

- You can gain citizenship if you have been living in a country continuously for at least p years;
- If you leave the country for q consecutive years without returning then your citizenship will be lost.

The values of p and q depend on the particular country.

You may assume that time spent travelling between countries is insignificant (i.e., zero), and that for each country, p will be an even integer number of years and q will be an odd integer number of years.

Example

Suppose you are interested in the five countries below:

Country	**Years to gain** (p)	**Years to lose** (q)
Australia	2	15
Brazil	6	3
France	6	11
India	4	7
Singapore	8	5

To gain simultaneous citizenship of as many different countries as possible, your travel plan might be:

- fly to France and remain there for 6 years to gain French citizenship;
- fly to Australia and remain there for 2 years to gain Australian citizenship;
- fly to India and remain there for 4 years to gain Indian citizenship;
- quickly stop in France to maintain your French citizenship (this takes no time);
- fly to Brazil for 6 years to gain citizenship of Brazil.

Note that if you had not revisited France after India, you would have lost French citizenship—the 2+4+6 years spent in Australia, India and Brazil would have exceeded the 11 year limit that France imposes.

This result is the best you can achieve—you cannot gain citizenship of Singapore, as you would lose citizenship of both Brazil and India in the process. There are many other possible

travel plans that could achieve four simultaneous citizenships, but no more.

Implementation

You should submit a file implementing the function *countries* as follows:

Your Function: `countries()`

C/C++
```
int countries(int N, int *P, int *Q);
```

Pascal
```
function countries(N : LongInt; var P, Q : array of LongInt) : LongInt;
```

Description

Your submission must implement this function.

This function should calculate the largest number of citizenships you can simultaneously hold.

Parameters

- ***N***: The number of countries.
- ***P***: An array of *N* integers that specify the number of years required to gain citizenship of each country.
- ***Q***: An array of *N* integers that specify the number of years of absence that will cause you to lose citizenship of each country.
- **Returns**: The largest number of citizenships you can simultaneously hold.

Sample Session

The following session describes the example above:

N	P	Q	Return value
5	[2, 6, 6, 4, 8]	[15, 3, 11, 7, 5]	4

Constraints

- Time limit: 1 second
- Memory limit: 32 MiB
- $1 \le N \le 100{,}000$
- $2 \le P[i] \le 10{,}000$, and *P[i]* is even
- $1 \le Q[i] \le 9{,}999$, and *Q[i]* is odd

Subtasks

Subtask	Points	Additional Input Constraints
1	30	You may assume $N \leq 100$
2	30	You may assume $N \leq 2500$
3	40	*(None)*

Experimentation

The sample grader provided with the task environment will read input from the file *citizen.in*, which must be in the following format:

- line 1: N
- line 2,. .., N+1: P[i] Q[i]

For instance, the example above should be provided in the following format:

```
5
2 15
6 3
6 11
4 7
8 5
```

Language Notes

C/C++ You must `#include "citizen.h"`.

Pascal You must define the `unit Citizen`. All arrays are numbered from 0 (not 1).

See the solution templates on your machine for examples.

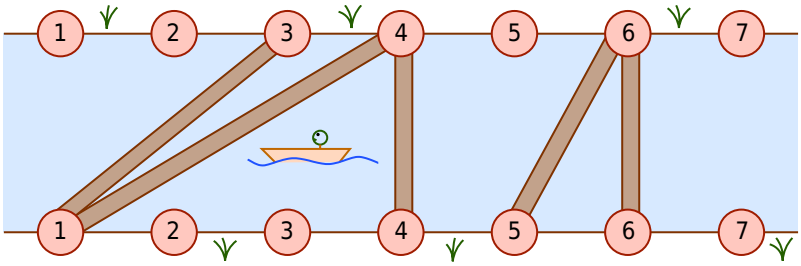
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Practice tasks

fog

English — 1.0

As Chief Surveyor of the island paradise of Oz, you are mapping the locations of the many bridges across the river. However, the entire island is covered in fog and you cannot see anything at all. Taking this as a challenge, you set out regardless in your trusty rowboat.

Your plan is to sail up and down the river, plotting bridges as you go—although you cannot see anything from the bank, you will see each bridge when you are directly beneath it. Because of the fog, you cannot steer accurately—you can only row in a straight line from one bank to another and count the bridges as you float beneath them.



The river has an upper bank and a lower bank. Each bank has N points along it numbered 1,2,...,N in order from left to right. This is illustrated above for N=7.

You do not know how many bridges there are across the river. However, you do know that:

- Each bridge starts at one of the points 1,2,...,N on the upper bank and ends at one of the points 1,2,...,N on the lower bank.
- No two bridges cross, although bridges may start and/or end at the same place.

Your task is to determine the number of bridges and their precise locations. You must do this by repeatedly rowing your boat from one bank to the other. Each time you do this:

- You must row your boat in a straight line between one of the points 1,2,...,N on the upper bank and one of the points 1,2,...,N on the lower bank.
- Each time you do this, you count the number of bridges that you float beneath. You do not count bridges that start or end at the same point as you on the river banks—you only count bridges that you float beneath while you are strictly between the two river banks.

Each line may start and end anywhere (i.e., you are not required to connect the next line to the previous line). Your arms are only strong enough to row in lines like this at most 250,000 times.

Implementation

You should submit a file that implements the procedure *goSail()*, using the function *row()* and the procedure *foundBridge()* that are provided by the grader. These functions and procedures are described below.

Grader Function: `row()`

C/C++
```
int row(int top, int bottom);
```
Pascal
```
function row(top, bottom: LongInt) : LongInt;
```

Description

Row your boat in a line between the upper bank and the lower bank.

The function returns the total number of bridges that you pass beneath. This count does not include bridges that end at *top* on the upper bank, or at *bottom* on the lower bank. If there is a bridge running directly from *top* to *bottom* (i.e., you remain beneath the bridge for the entire time), then this function will return −1.

Parameters

- ***top***: The point on the upper bank where you start rowing.
- ***bottom***: The point on the lower bank where you finish rowing.
- **Returns**: The number of bridges you pass beneath, or -1 if you remain under a bridge the entire time.

Grader Procedure: `foundBridge()`

C/C++
```
void foundBridge(int top, int bottom);
```
Pascal
```
procedure foundBridge(top, bottom : LongInt);
```

Description

Call this procedure when you have identified the location of a bridge.

Parameters

- ***top***: The point on the upper bank where the bridge ends.
- ***bottom***: The point on the lower bank where the bridge ends.

Your Procedure: `goSail()`

C/C++
```
void goSail(int N);
```
Pascal
```
procedure goSail(N: longint);
```

Description

Your submission must implement this procedure.

This function should use the provided grader routines `row()` and `foundBridge()` to determine the location of the bridges.

This function may call `row()` at most 250,000 times. It must call `foundBridge()` to report

the locations of the bridges, *in the order the bridges occur along the river from left to right*.

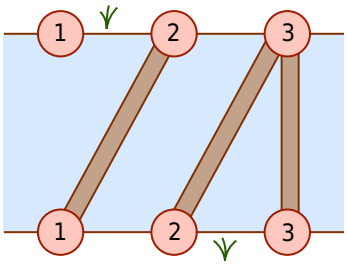
All arguments *top* and *bottom* to the functions `row()` and `foundBridge()` must be integers in the range $1 \le top,bottom \le N$. You may assume that `row()` runs in constant time.

Parameters

- ***N***: The number of points along each bank of the river.

Sample Session

Suppose the river and bridges are arranged according to the diagram below:



A sample session might run as follows. Assume your function *goSail* is called with the argument $N = 3$, indicating that there are three points on each river bank.

Function Call	Returns	Explanation
row(1,1)	0	The path from point 1 on the upper bank to point 1 on the lower bank does not pass beneath any bridges. The leftmost bridge, which starts at point 1 on the lower bank, is not counted.
row(2,2)	0	The path from point 2 on the upper bank to point 2 on the lower bank does not pass beneath any bridges.
row(3,3)	−1	There is a bridge running directly from point 3 on the upper bank to point 3 on the lower bank.
row(1,2)	1	The path from point 1 on the upper bank to point 2 on the lower bank passes beneath one bridge (this is the leftmost bridge).
row(3,2)	−1	There is a bridge running directly from point 3 on the upper bank to point 2 on the lower bank.
foundBridge(2,1)		You claim to have found a bridge between point 2 on the upper bank to point 1 on the lower bank.
foundBridge(3,2)		You claim to have found a bridge between point 3 on the upper bank to point 2 on the lower bank.
foundBridge(3,3)		You claim to have found a bridge between point 3 on the upper bank to point 3 on the lower bank.

This sample session finds the bridges correctly and reports them in the correct order from left to right.

Constraints

- Time limit: 1 second
- Memory limit: 32 MiB
- $1 \le N \le 100{,}000$

Subtasks

Subtask	Points	Additional Input Constraints
1	30	You may assume $N \le 600$
2	35	You may assume $N \le 15{,}000$
3	35	*(None)*

Experimentation

The sample grader provided with the task environment will read input from the file *fog.in*, which must be in the following format:

- line 1: N
- line 2: B
- line 3,...,B+2: Top[i] Bottom[i]

Here B is the number of bridges, and the *i*th bridge runs from point Top[i] on the upper bank to point Bottom[i] on the lower bank. The bridges must appear in this file in order from left to right.

For instance, the example above would be provided in the following format:

```
3
3
2 1
3 2
3 3
```

Language Notes

C/C++	You must `#include "fog.h"`.
Pascal	You must define the `unit Fog`, and you must also import the grader routines via `uses GraderHelpLib`. All arrays are numbered from 0 (not 1).

See the solution templates on your machine for examples.