TASK OVERVIEW SHEET / DAY-2

TASK		BATCH	BUS	RODS
Task material directory	Linux	~/batch	~/bus	~/rods
	WinXP	C:\IOI\batch	C:\IOI\bus	C:\IOI\rods
Time limit per test		0.1 secs	4 secs	1 sec
Memory limit		32 MB	32 MB	32 MB
Compiler options	C and C++	-O2 -static -lm	-O2 -static -lm	-O2 -static -lm
	Pascal	-So -O2 -XS	-So -O2 -XS	-So -O2 -XS
Number of tests		20	20	20
Maximum points per test		5	5	5
Maximum total points		100	100	100
Program header comment when using C		/* TASK: batch LANG: C */	/* TASK: bus LANG: C */	/* TASK: rods LANG: C */
Program header comment when using C++		/* TASK: batch LANG: C++ */	/* TASK: bus LANG: C++ */	/* TASK: rods LANG: C++ */

Program header comment when using Pascal		{
		TASK: batch
		LANG: PASCAL
		}
			{
			TASK: bus
			LANG: PASCAL
			}
				{
				TASK: rods
				LANG: PASCAL
				}

Submission is accepted, if;		Example 1 is solved.	Example 1 is solved.	Example is processed.

Batch Scheduling

PROBLEM

There is a sequence of N jobs to be processed on one machine. The jobs are numbered from 1 to N, so that the sequence is 1,2, …, N. The sequence of jobs must be partitioned into one or more batches, where each batch consists of consecutive jobs in the sequence. The processing starts at time 0. The batches are handled one by one starting from the first batch as follows. If a batch b contains jobs with smaller numbers than batch c, then batch b is handled before batch c. The jobs in a batch are processed successively on the machine. Immediately after all the jobs in a batch are processed, the machine outputs the results of all the jobs in that batch. The output time of a job j is the time when the batch containing j finishes.

A setup time S is needed to set up the machine for each batch. For each job i, we know its cost factor F_i and the time T_i required to process it. If a batch contains the jobs x, x+1, … , x+k, and starts at time t, then the output time of every job in that batch is $t + S + (T_x + T_{x+1} + \ldots + T_{x+k})$. Note that the machine outputs the results of all jobs in a batch at the same time. If the output time of job i is O_i, its cost is $O_i \times F_i$. For example, assume that there are 5 jobs, the setup time $S = 1$, $(T_1, T_2, T_3, T_4, T_5) = (1, 3, 4, 2, 1)$, and $(F_1, F_2, F_3, F_4, F_5) = (3, 2, 3, 3, 4)$. If the jobs are partitioned into three batches {1, 2}, {3}, {4, 5}, then the output times $(O_1, O_2, O_3, O_4, O_5) = (5, 5, 10, 14, 14)$ and the costs of the jobs are (15, 10, 30, 42, 56), respectively. The total cost for a partitioning is the sum of the costs of all jobs. The total cost for the example partitioning above is 153.

You are to write a program which, given the batch setup time and a sequence of jobs with their processing times and cost factors, computes the minimum possible total cost.

INPUT

Your program reads from standard input. The first line contains the number of jobs N, $1 \le N \le 10000$. The second line contains the batch setup time S which is an integer, $0 \le S \le 50$. The following N lines contain information about the jobs 1, 2, …, N in that order as follows. First on each of these lines is an integer T_i, $1 \le T_i \le 100$, the processing time of the job. Following that, there is an integer F_i, $1 \le F_i \le 100$, the cost factor of the job.

OUTPUT

Your program writes to standard output. The output contains one line, which contains one integer: the minimum possible total cost.

EXAMPLE INPUTS AND OUTPUTS

Example 1: input

```
2
50
100 100
100 100
```

output

```
45000
```

Example 2: input

```
5
1
1 3
3 2
4 3
2 3
1 4
```

output

```
153
```

Example 2 is the example in the text.

REMARK

For each test case, the total cost for any partitioning does not exceed $2^{31} - 1$.

SCORING

If your program outputs the correct answer for a test case within the time limit, then you get full points for the test case, and otherwise you get 0 points.

Bus Terminals

PROBLEM

Yong-In city plans to build a bus network with N bus stops. Each bus stop is at a street corner. Yong-In is a modern city, so its map is a grid of square blocks of equal size. Two of these bus stops are to be selected as hubs H_1 and H_2. The hubs will be connected to each other by a direct bus line and each of the remaining $N - 2$ bus stops will be connected directly to either H_1 or H_2 (but not to both), but not to any other bus stop.

The distance between any two bus stops is the length of the shortest possible route following the streets. That is, if a bus stop is represented as (x, y) with x-coordinate x and y-coordinate y, then the distance between two bus stops (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$. If bus stops A and B are connected to the same hub H_1, then the length of the route from A to B is the sum of the distances from A to H_1 and from H_1 to B. If bus stops A and B are connected to different hubs, e.g., A to H_1 and B to H_2, then the length of the route from A to B is the sum of the distances from A to H_1, from H_1 to H_2, and from H_2 to B.

The planning authority of Yong-In city would like to make sure that every citizen can reach every point within the city as quickly as possible. Therefore, city planners want to choose two bus stops to be hubs in such a way that in the resulting bus network the length of the longest route between any two bus stops is as short as possible.

One choice P of two hubs and assignments of bus stops to those hubs is better than another choice Q if the length of the longest bus route is shorter in P than in Q. Your task is to write a program to compute the length of this longest route for the best choice P.

INPUT

Your program is to read from standard input. The first line contains one positive integer N, $2 \le N \le 500$, the number of bus stops. Each of the remaining N lines contains the x-coordinate followed by the y-coordinate of a bus stop. The x- and y-coordinates are positive integers ≤ 5000. No two bus stops are at the same location.

OUTPUT

Your program is to write to standard output. The output contains one line with a single positive integer, the minimum length of the longest bus route for the input.

EXAMPLE INPUTS AND OUTPUTS

Example 1:

input

```
6
1 7
16 6
12 4
4 4
1 1
11 1
```

output

```
20
```

Example 2:

input

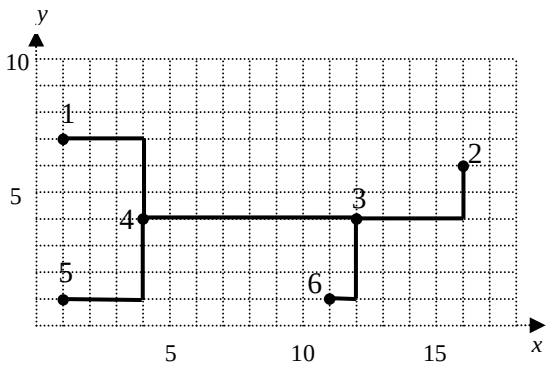
```
7
7 9
10 9
5 3
1 1
7 2
15 6
17 7
```

output

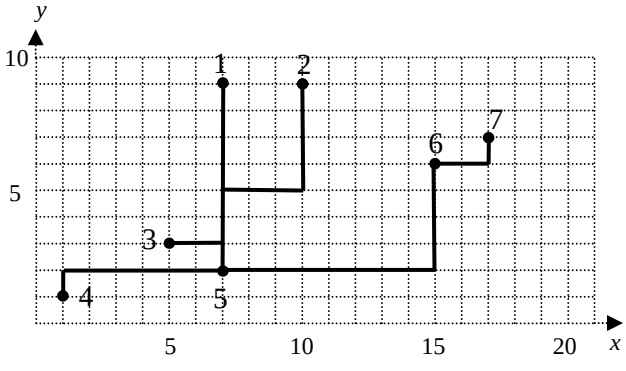
```
25
```

The following figures show the bus networks for the inputs given above. If in Example 1 bus stops 3 and 4 are selected as hubs then the longest route is either between bus stops 2 and 5 or between bus stops 2 and 1. There is no better choice for the hubs, and the answer is 20.

For the bus network in Example 2, if bus stops 5 and 6 are selected as hubs then the longest route is obtained between bus stops 2 and 7. There is no better choice for the hubs, and the answer is 25.



Bus network for Example 1



Bus network for Example 2

SCORING

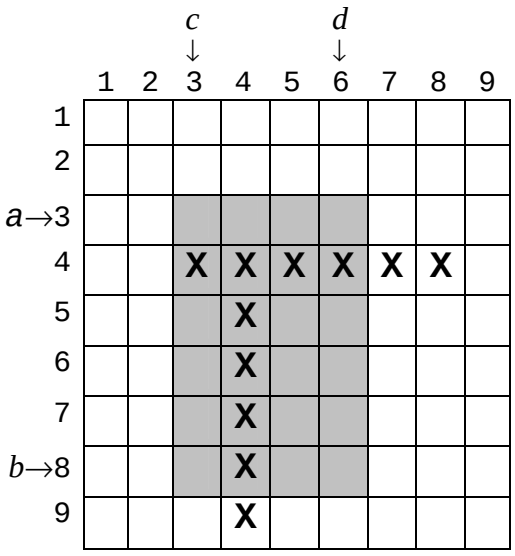
If your program outputs the correct answer for a test case within the time limit, then you get full points for that test case, and otherwise you get 0 points for that case.

Two Rods

PROBLEM

A rod is either a horizontal or a vertical sequence of at least 2 consecutive grid cells. Two rods, one horizontal and the other vertical, are placed on an N by N grid. In Figure-1, the two rods are shown by **X**'s. The rods may or may not be the same length; furthermore, they may share a cell. If, from a diagram such as Figure-1, it is possible to interpret a cell, e.g. (4,4), as being in just one rod or in both rods, we make the interpretation that the cell is in both. Hence, the top cell of the vertical rod is (4,4) rather than (5,4).

Figure-1



Initially we do not know where the two rods are, and so your task is to write a program to determine their locations. We call the horizontal rod ROD1, and the vertical rod ROD2. Each grid cell is represented by a row/column pair (r,c), and the top left corner of the grid is taken to be location (1,1). Each rod is represented as two cells, $<(r_1, c_1), (r_2, c_2)>$. In Figure-1 ROD1 is <(4,3), (4,8)> and ROD2 is <(4,4), (9,4)>.

This task involves the use of library functions for input, for determining the solution, and for output. The length of a side of the square grid is given by the library function `gridsize`, which your program is to call at the beginning of each test case. To locate the rods, you can only use the library function `rect(a,b,c,d)`, which examines the rectangular region $[a,b]\times[c,d]$ (shaded region in Figure-1), where $a \le b$ and $c \le d$. [Note carefully the order of these parameters.] If at least one grid cell of either rod falls inside the query rectangle $[a,b]\times[c,d]$, `rect` returns 1; otherwise it returns 0. So in the example, `rect(3,8,3,6)`returns `1`. Your task is to write a program to discover the exact location of the rods using a limited number of `rect` calls.

You produce output by calling another library function `report(`$r_1, c_1, r_2, c_2, p_1, q_1, p_2, q_2$`)` where ROD1 is $<(r_1, c_1),(r_2, c_2)>$ and ROD2 is $<(p_1, q_1),(p_2, q_2)>$. Calling `report` terminates your program. Recall that ROD1 is horizontal and ROD2 is vertical, and (r_1,

c_1) is the left end cell of the horizontal rod ROD1. Cell (p_1, q_1) is the top end cell of ROD2. Hence $r_1 = r_2$, $c_1 < c_2$, $p_1 < p_2$, and $q_1 = q_2$. If your `report` parameters do not meet these constraints, then you will get error messages on standard output.

CONSTRAINTS

- You can access input only by using the library functions `gridsize` and `rect`.
- N, the maximum row (column) size of input, satisfies $5 \le N \le 10000$.
- The number of `rect` calls should be at most 400 for every test case. If your program calls `rect` more than 400 times, this will terminate your program.
- Your program must call `rect` more than once and call `report` exactly once.
- If a `rect` call is not valid (e.g., the query range exceeds the grid space), it will terminate your program.
- Your program must not read or write any files and must not use any standard input/output.

LIBRARY

You are given a library in the following:

FreePascal Library (`prectlib.ppu`, `prectlib.o`)

```
function gridsize: LongInt;
function rect(a,b,c,d : LongInt) : LongInt;
procedure report(r1, c1, r2, c2, p1, q1, p2, q2 : LongInt);
```

Instructions: To compile your `rods.pas`, include the import statement

```
uses prectlib;
```

in the source code and compile it as

```
fpc -So -O2 -XS rods.pas
```

The program `prodstool.pas` gives an example of using this FreePascal library.

GNU C/C++ Library (`crectlib.h`, `crectlib.o`)

```
int gridsize();
int rect(int a, int b, int c, int d);
void report(int r1, int c1, int r2, int c2, int p1, int q1,
            int p2, int q2);
```

Instructions: To compile your rods.c, use

```
#include "crectlib.h"
```

in the source code and compile it as:

```
gcc -O2 -static rods.c crectlib.o -lm
g++ -O2 -static rods.cpp crectlib.o -lm
```

The program `crodstool.c` gives an example of using this GNU C/C++ library.

For C/C++ in the RHIDE environment

Be sure that you set the Option->Linker configuration to `crectlib.o` .

EXPERIMENTATION

To experiment with the library, you must create a text file `rods.in`. The file must contain three lines. The first line contains one integer: N, the size of the grid. The second line contains the coordinates of ROD1, r_1 c_1 r_2 c_2; where (r_1, c_1) is the left end cell of ROD1. The third line contains the coordinates of ROD2, p_1 q_1 p_2 q_2, where (p_1, q_1) is the top end cell of ROD2.

After running your program which calls `report`, you will get the output file `rods.out`. This file contains the number of `rect` function calls and the coordinates of the ends of the rods you submitted in your call to `report`. If there are any errors or violations of the requirements during library calls, then `rods.out` will contain the corresponding error messages.

The dialogue between your program and the library is recorded in the file `rods.log`. This log file `rods.log` shows the sequence of function calls your program made in the form of "*k* : `rect(`*a*`,`*b*`,`*c*`,`*d*`)` = *ans*", which means *k*-th function call `rect(`*a*`,`*b*`,`*c*`,`*d*`)` returns *ans*.

EXAMPLE INPUT AND OUTPUT

Example: rods.in

```
9
4 3 4 8
4 4 9 4
```

rods.out

```
20
4 3 4 8
4 4 9 4
```

SCORING

If your program violates any of the constraints (e.g., more than 400 `rect` calls), or if your program's output (the locations of the rods) is not correct, the score is 0.

If your program's output is correct, then your score depends on the number of `rect` calls for each testing data. For each test case if the number of `rect` calls is at most 100, then you get 5 points. If your program calls `rect` 101 to 200 times, you get 3 points. If the number of `rect` calls is between 201 and 400, then you get 1 point.