problem	kasino	polja

source file	`kasino.pas`
	`kasino.c`
	`kasino.cpp`
		`polja.pas`
		`polja.c`
		`polja.cpp`

input data	library	stdin
output data	library	stdout
time limit (intel celeron 2.4 ghz)	1 sec	2 sec
memory limit (heap)	64 MB	64 MB
memory limit (stack)	16 MB	16 MB
points	**100**	**100**
	200	

kasino

Shuffling machine in one casino works in the following way:

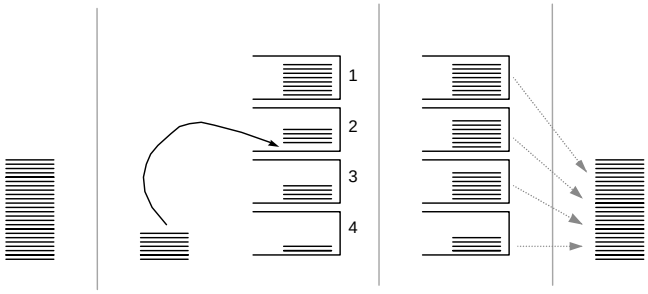
Deck consists of N different cards denoted by numbers from 1 to N ($1 \leq N \leq 1000$), and the machine consists of K ($1 \leq K \leq 5$) compartments denoted by numbers from 1 to K.

In the beginning, cards are sorted in the ascending order i.e. card with number 1 is on the top of the deck and card with number N is at the bottom.

In each step, machine chooses 2 **random** numbers A and B, $1 \leq A \leq K$, $1 \leq B \leq 2$, and takes the card from the top of the deck and puts it in the compartment A. If B is equal to 1, then it puts the card on the top of cards in compartment A, and if B is equal to 2, then it puts the card on the bottom of cards in compartment A.

After all the cards are moved from the original deck to compartments, compartments are merged. Cards from the first compartment are going to the top, cards from the second compartment under them, and so on...

Example of shuffling:



Now, Mike and Jack are entering the casino. They are taking a deck with shuffled cards and they are playing an interesting game.

Mike takes the cards, and Jack plays the game of guessing cards that consists of N steps.

In each step, Jack tries to guess which card is at some position in the deck. More precisely, Jack says two numbers P and R, $1 \leq P,R \leq N$, and then Mike says which card is on the position P (card on the top of the deck is on the position 1, and card at the bottom is on the position N). If at the position P there is a card with number R, Jack **gets 5 dollars**, otherwise he **looses 1 dollar**. For each position i.e. number between 1 and N, Jack must ask **exactly once**.

At the beginning of the game, Jack has **0 dollars**, and it is allowed that during the game this amount of money goes below zero.

Write a program that will play this game by calling functions from the library, in such a way that at the end of the game the amount of dollars Jack has **is a positive number**.

library

You are given a library **`kasinolib`** that includes two functions:

`Init` - you should call this function exactly once at the beginning of your program. It returns values N and K.

```
procedure Init(var n,k : integer);
void Init(int *n,int *k);
```

`Igraj` - this function is to be called with two arguments - numbers P i R.

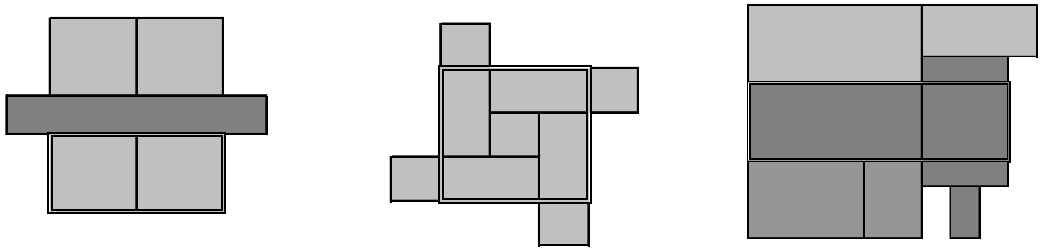
```
procedure Igraj(p,r : integer);
int Igraj(int p,int r);
```

polja

Mickey is in the airplane above ground. On the ground, he sees fields of rectangular shapes and different colors, depending on planted vegetation.

Mickey wants to know what is the **area of the largest one-colored rectangle** on the ground.

On the picture you can see examples from the bottom of the page (solution for each example is marked):



Write a program that will calculate the area of the largest one-colored rectangle on the ground.

Two different fields **will not overlap**, however they may touch in corners or along sides.

input data

First line of input contains an integer N, $1 \leq N \leq 2500$, the number of fields on the ground. Each of the following N lines contains five integers X1, Y1, X2, Y2 (X1<X2, Y1<Y2) and C.

These numbers represents one field (with sides parallel to coordinate axes) with coordinates of diagonal corners (X1,Y1) and (X2,Y2) and with color C, $1 \leq C \leq 100$. All the coordinates will be integers between 0 and 1,000,000,000 (inclusive).

output data

First and only line of output should contain the area from the problem statement.

Note: solution will always fit into the signed 64-bit integer data type (int64 in Pascal, long long in C/C++).

examples

input

```
5
1 1 3 3 1
3 1 5 3 1
1 4 3 6 1
3 4 5 6 1
0 3 6 4 2
```

output

```
8
```

input

```
5
5 5 6 6 22
3 4 6 5 22
6 3 7 6 22
5 6 8 7 22
4 5 5 8 22
```

output

```
9
```

input

```
7
0 0 4 3 1
6 2 9 7 2
6 7 10 9 3
4 0 6 3 1
0 6 6 9 3
0 3 6 6 2
7 0 8 2 2
```

output

```
27
```