2007 Canadian Computing Competition
Day 1, Question 1

Problem A – Cows

Input file: `cows.in`
Output: to standard output
Source file: `cows.{c, cpp, pas}`

Your friend to the south is interested in building fences and turning plowshares into swords. In order to help with his overseas adventure, they are forced to save money on buying fence posts by using trees as fence posts wherever possible. Given the locations of some trees, you are to help farmers try to create the largest pasture that is possible. Not all the trees will need to be used.

However, because you will oversee the construction of the pasture yourself, all the farmers want to know is how many cows they can put in the pasture. It is well known that a cow needs at least 50 square metres of pasture to survive.

Input

The first line of input contains a single integer, n ($1 \leq n \leq 10000$), containing the number of trees that grow on the available land. The next n lines contain the integer coordinates of each tree given as two integers x and y separated by one space (where $-1000 \leq x, y \leq 1000$). The integer coordinates correlate exactly to distance in metres (e.g., the distance between coordinate $(10, 11)$ and $(11, 11)$ is one metre).

Output

You are to output a single integer value, the number of cows that can survive on the largest field you can construct using the available trees.

Sample Input

```
4
0 0
0 101
75 0
75 101
```

Output for Sample Input

```
151
```

2007 Canadian Computing Competition
Day 1, Question 2

Problem B – Snowflakes

Input file: `snowflakes.in`
Output: to standard output
Source file: `snowflakes.{c, cpp, pas}`

You may have heard that no two snowflakes are alike. Your task is to write a program to determine whether this is really true. Your program will read information about a collection of snowflakes, and search for a pair that may be identical.

Each snowflake has six arms. For each snowflake, your program will be provided with a measurement of the length of each of the six arms. Any pair of snowflakes which have the same lengths of corresponding arms should be flagged by your program as possibly identical.

Input

The first line of input will contain a single integer $n, 0 < n \leq 100000$, the number of snowflakes to follow. This will be followed by n lines, each describing a snowflake. Each snowflake will be described by a line containing six integers (each integer is at least 0 and less than 10000000), the lengths of the arms of the snowflake. The lengths of the arms will be given in order around the snowflake (either clockwise or counterclockwise), but they may begin with any of the six arms. For example, the same snowflake could be described as 1 2 3 4 5 6 or 4 3 2 1 6 5.

Output

If all of the snowflakes are distinct, your program should print the message:
`No two snowflakes are alike.`
If there is a pair of possibly identical snowflakes, your program should print the message:
`Twin snowflakes found.`

Sample Input

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

Output for Sample Input

```
Twin snowflakes found.
```

2007 Canadian Computing Competition
Day 1, Question 3

Problem C – Bowling for Numbers++

Input file: `bowling.in`
Output: to standard output
Source file: `bowling.{c, cpp, pas}`

Recall that at the Canadian Carnival Competition (CCC), a popular game was *Bowling for Numbers*. A large number of bowling pins were lined up in a row. Each bowling pin had a number printed on it, which was the score obtained from knocking over that pin. The player was given a number of bowling balls; each bowling ball was wide enough to knock over a few consecutive and adjacent pins.

For example, one possible sequence of pins was:

```
2 8 5 1 9 6 9 3 2
```

If Alice was given two balls, each able to knock over three adjacent pins, the maximum score Alice could achieve would be 39, the sum of two throws: 2+8+5 = 15, and 9+6+9 = 24.

Since many, including you, have mastered the art of throwing balls at pins, the old and wise members of the Canadian Carnival Competition Committee have decided to make the game slightly more difficult by introducing the concept of *penalty pins*.

Penalty pins are pins with a *negative score*, such that a player's score *decreases* when they are knocked over. This can change the player's strategy, as the player can use the empty spaces to the left and right of the pins, as well as spaces created by previous throws, to avoid hitting penalty pins. Consider the following example:

```
2 8 -5 3 5 8 4 8 -6
```

If Alice was given three balls, each able to knock over three adjacent pins, the maximum score Alice could achieve would be 38, the sum of three throws: $2 + 8$, $3 + 5 + 8$, and $4 + 8$. Alice's first throw is deliberately to the left and catches only the leftmost two pins, avoiding the `-5`. Alice's second throw hits `3 5 8`, and her third throw catches the remaining `4 8`, going through the space created by her second throw to avoid the `-6`.

Bob has a strategy where he picks the shot that gives him the most score, then repeatedly picks the shot that gives him the most score from the remaining pins. This strategy doesn't always yield the maximum score, but is close. On the test data, such a strategy would get a score of 20%.

Input

Input consists of a series of test cases. The first line of input is t, $1 <= t <= 10$, indicating the number of test cases in the file.

The first line of each test case contains three integers. First is the integer n, $1 <= n <= 10000$, indicating the number of bowling pins. The second integer, k, $1 <= k <= 500$, giving the number of bowling balls available to each player. The third and final integer is w, $1 <= w <= 100$, the width of the bowling ball (the number of adjacent pins it can knock over).

The next n lines of each test case each contain a single integer giving the score of the pins, in order. The scores are in the range $-10000...10000$ inclusive.

20% of the test data will have size $n \leq 50$.

Output

For each test case, output the maximum achievable score by the player. This score is guaranteed to be less than one billion.

Sample Input

```
2
9 2 3
2
8
5
1
9
6
9
3
2
9 3 3
2
8
-5
3
5
8
4
8
-6
```

Sample Output

```
39
38
```