Exercises

- For all the exercises, load a file as input and return the answer printed in console.
- You can use the programming language of your preference.

1. For a Array with n-arrays each one with 3 numbers that represent a triangle sides, return an array with all the arrays that represent a right triangle.

Example: input -> [[1,3,5],[6,8,10],[5,14,8],[7,24,25].....]
output -> [[6,8,10],[7,24,25]....]

2. Given a string with n characters, find the longest substring that is the same in reverse.

Example: input -> "I like racecars that go fast"
output -> "racecar"

3. Write a function, where given a string of arbitrary characters, returns true if all brackets (defined as parentheses, square-brackets, curly-braces, and chevrons) are correctly paired and ordered. This is to say that all brackets, if they enclose other brackets, enclose both the paired opening and closing characters.

Examples: input -> "([<{abc123abc}>])"
output -> true

input -> "(abc[123)abc]"
output -> false

4.Your task will be to write a program that will be able to find topics or questions that are near to a given input location, up to a specified limit.

Input format:

- The first line of input will be 3 integers: number of topics T, number of questions Q, and number of queries N.
- There will be T lines following that, each with a topic id integer, and two doubles representing that topic's location (you can consider the points to be located on a XY plane, location of a entity is in form of its x and y coordinate in the plane).
- There will be Q lines following that, each with a question id integer and number of topics the question is associated with, Qn, followed by Qn number of topic ids associated with that question. These integers are guaranteed to be well-formed existing topics. There may be 0 topics associated with a question, and in such cases, the question should never appear in the query results.
- The next N lines will consist of a char, an integer and two doubles each, representing the type of results required ("t" for topic or "q" for question), the number of results required, and the location to be used as the query.

Sample Input:

```
3 6 2
0 0.0 0.0
1 1.0 1.0
2 2.0 2.0
0 1 0
1 2 0 1
2 3 0 1 2
3 0
4 0
5 2 1 2
t 2 0.0 0.0
q 5 100.0 100.0
```

Output format:

For each query line given in the input, you are to output the ids of the nearest entities in ascending order of distance from the query location, up to the specified number of results. When there is a tie between different entities in the ordering due to equal distances (threshold of 0.001 for equality comparisons), the entity with the larger id should be ranked first.
Distance of a question from a point is the minimum of the distance of all topics associated with that question.

Sample Output:

```
0 1
5 2 1 0
```

Explanation:

There are 3 topics with ids 0, 1 and 2. There are also 6 questions, with ids 0 to 5. We first ask a nearest topic query.
The closest 2 topics to (0.0, 0.0) are topics 0 and 1.
The next query asks for upto 5 closest questions to (100.0, 100.0). Since questions 5 and 2 are tagged with topic 2 located at (2.0, 2.0), they are closest to the query location.
Because of the tie in distance, we put question 5 before question 2.
The next closest question is question 1, followed by question 0.
We do not output questions 3 or 4 because there are no topics associated with them.

Constraints:

1 <= T <= 10000
1 <= Q <= 1000
1 <= N <= 10000
Integer ids are between 0 and 100000 inclusive.
Number of topics associated with a question is not more than 10.
The number of results required for a query is not more than 100.
0.0 <= x,y <= 1000000.0 (10^6)

For the large testcases, all topic x,y co-ordinates will be approximately uniformly distributed over the bounds.