Preparation (before the test):

Test assignment

Thank you for participating in our recruiting test. This will be a JavaScript programming test!

How to prepare for this test

Install an editor and a modern JavaScript-enabled browser of your choice. It is not permitted to use any 3rd-party libraries or dependencies. However you will not need them anyway.

The task is a very general programming assignment testing general problem structuring and programming proficiency. Take the time you need to develop your solution, but you should submit within 24 hours after receiving the test. We leave it to the candidates to test their solutions thoroughly before handing them in.

You must develop the solution yourself. You may not let others help you or search for existing solutions. Of course, you may use any documentation of the JavaScript language. Do not give your solution to others or make it public. It may entice others to send in plagiarized solutions, and thus make our recruiting less fair. Publishing the task description is also considered cheating and will void your application.

The amount of code you need for a correct solution is relatively small, and you will spend most of the allocated time thinking rather than typing. We would like to receive the best solution you can come up with!

Please note that disclosing any details of our test assignment or interview questions can lead to **serious consequences** for your own candidacy and that of future applicants. Please be so kind as to read and confirm the following statements:

- You have not participated in a think-cell recruiting test before; and

- You understand the confidential nature of think-cell's application process. You will not reveal or publish any of the tasks or questions presented to you by think-cell. You **will not share information about these tasks or questions with any third party**, be it in public or private.

Task description:

You are given the following source code:

```
const sampleInput = [
     {
         key: [1, 4],
         val: 'red'
     },
     {
         key: [-2, -1],
         val: 'green'
     },
     {
         key: [2, 40],
         val: 'blue'
     }
];

function makeQuery(input) {
     return key => input.find(function(item) {
         return item.key[0]<=key &&
key<item.key[1];
     })?.val;
}
```

The array `sampleInput` contains a list of intervals (`key`) which are mapped to a value (`val`). The function `makeQuery` takes input of this format and returns a query function. The query function takes a `key` argument and finds the interval that (right exclusive) contains this key. It then returns the corresponding value for this interval. If no interval is found, the query function returns “undefined”. The function can be used like this:

```
query=makeQuery(sampleInput);
query(3); // returns 'red'
```

Note that `sampleInput` is just example data. You should expect the real input to be much bigger, and it can contain arbitrary data for mapped values. The input is not ordered, and the first interval takes precedence in case of overlaps.

Your task is to rewrite and improve `makeQuery`. The provided query function is very simple and inefficient for large input data. Imagine `makeQuery` to be called once on a given input and preprocess it. Your returned query function should be able to efficiently handle many requests for arbitrary input data. Apart from that it should give the same results as our query function.