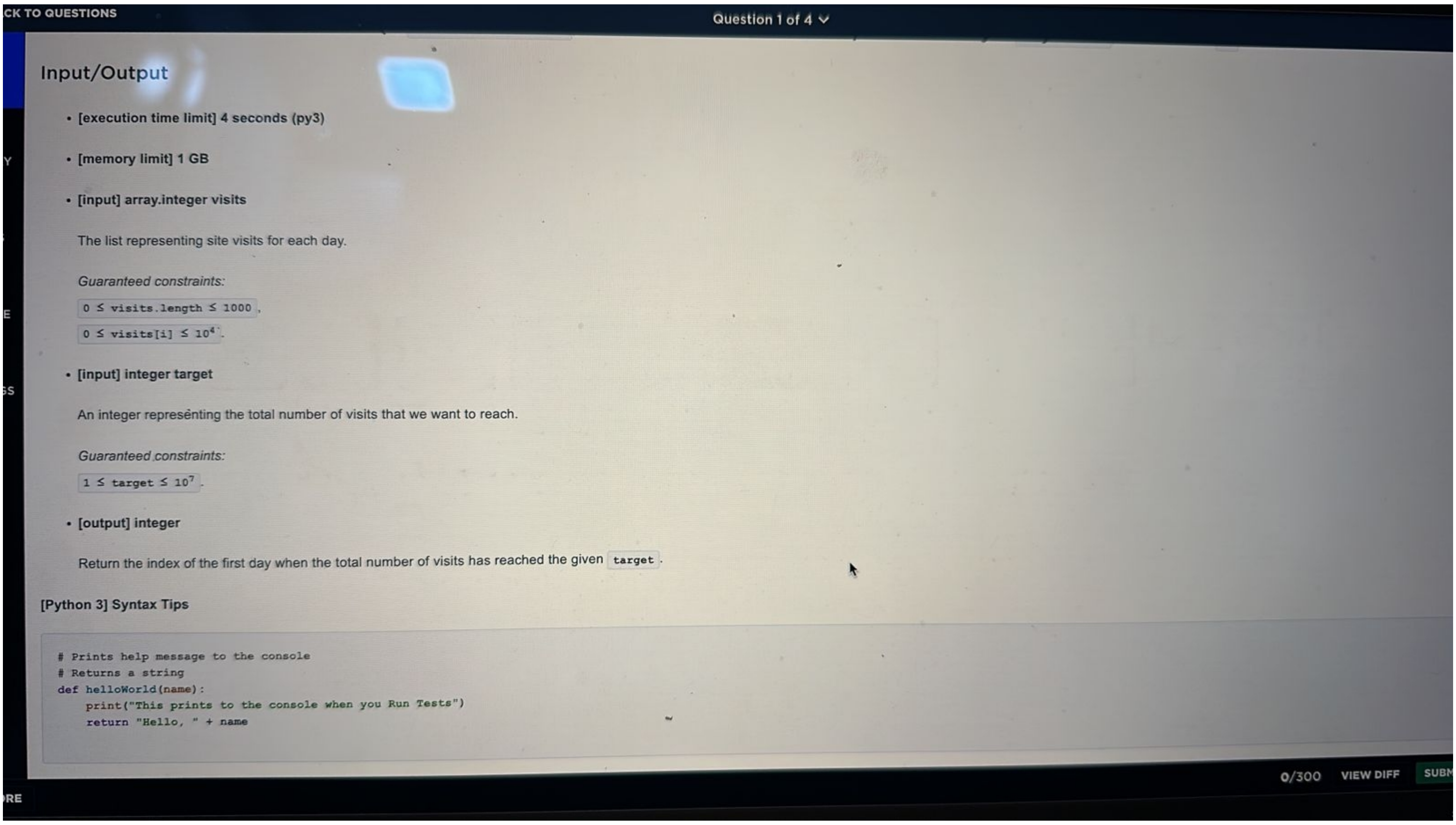


Input/Output

- **[execution time limit] 4 seconds (py3)**
- **[memory limit] 1 GB**
- **[input] array.integer visits**

  The list representing site visits for each day.

  Guaranteed constraints:
  $0 \leq visits.length \leq 1000$,
  $0 \leq visits[i] \leq 10^4$.

- **[input] integer target**

  An integer representing the total number of visits that we want to reach.

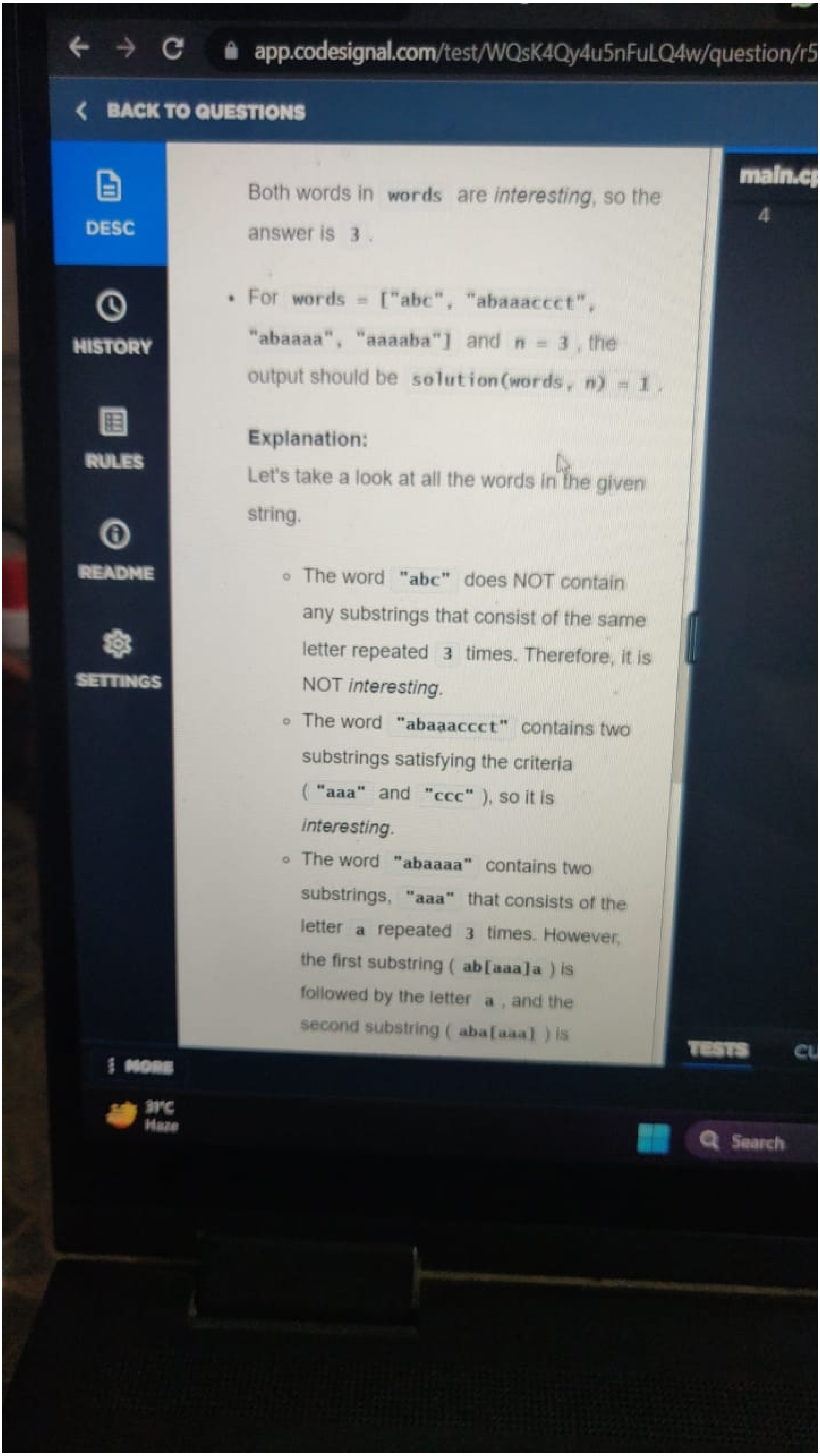
  Guaranteed constraints:
  $1 \leq target \leq 10^7$.

- **[output] integer**

  Return the index of the first day when the total number of visits has reached the given `target`.

[Python 3] Syntax Tips

```
# Prints help message to the console
# Returns a string
def helloWorld(name):
    print("This prints to the console when you Run Tests")
    return "Hello, " + name
```



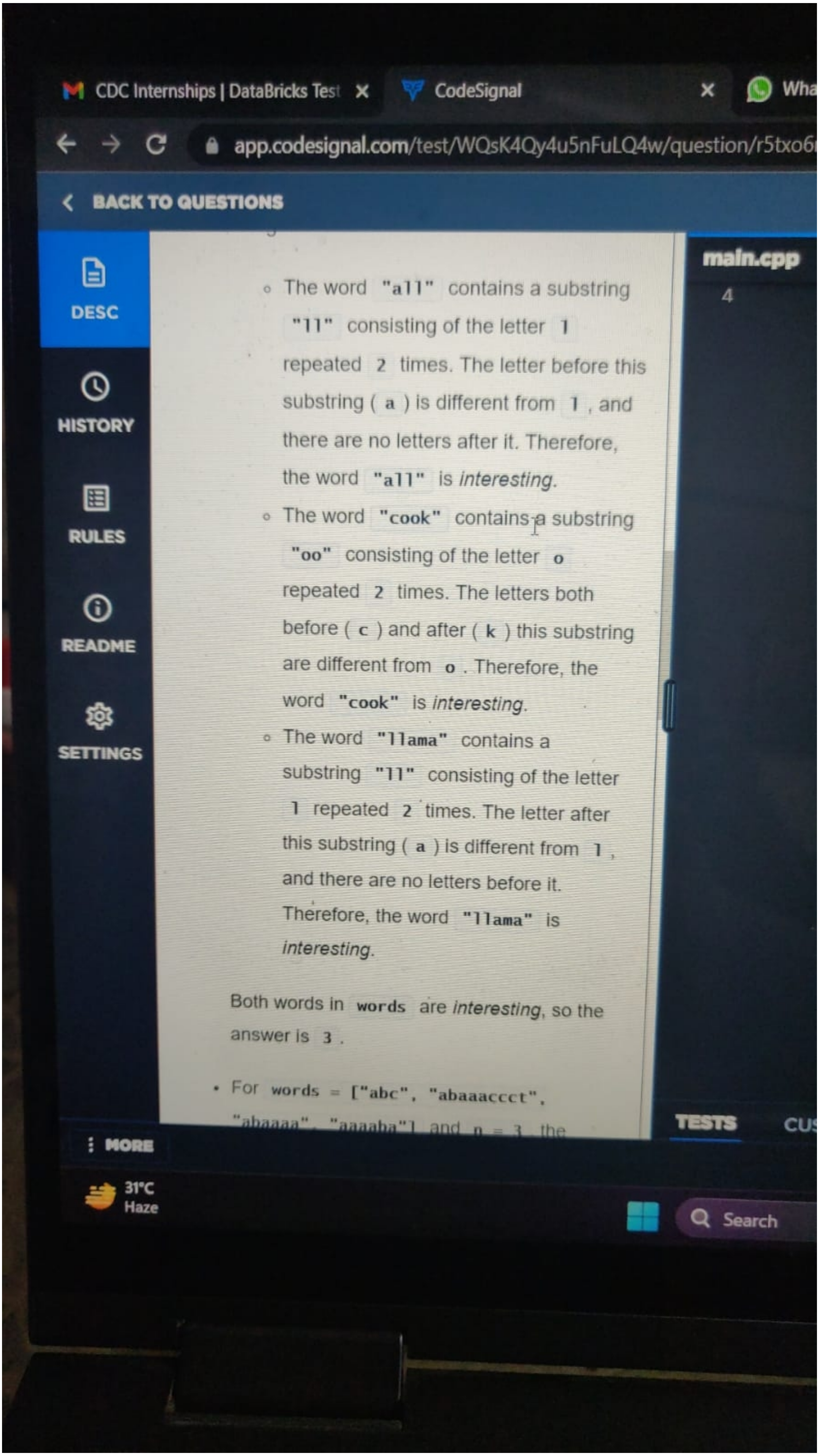
Both words in `words` are *interesting*, so the answer is `3`.

- For `words = ["abc", "abaaaccct", "abaaaa", "aaaaba"]` and `n = 3`, the output should be `solution(words, n) = 1`.

Explanation:

Let's take a look at all the words in the given string.

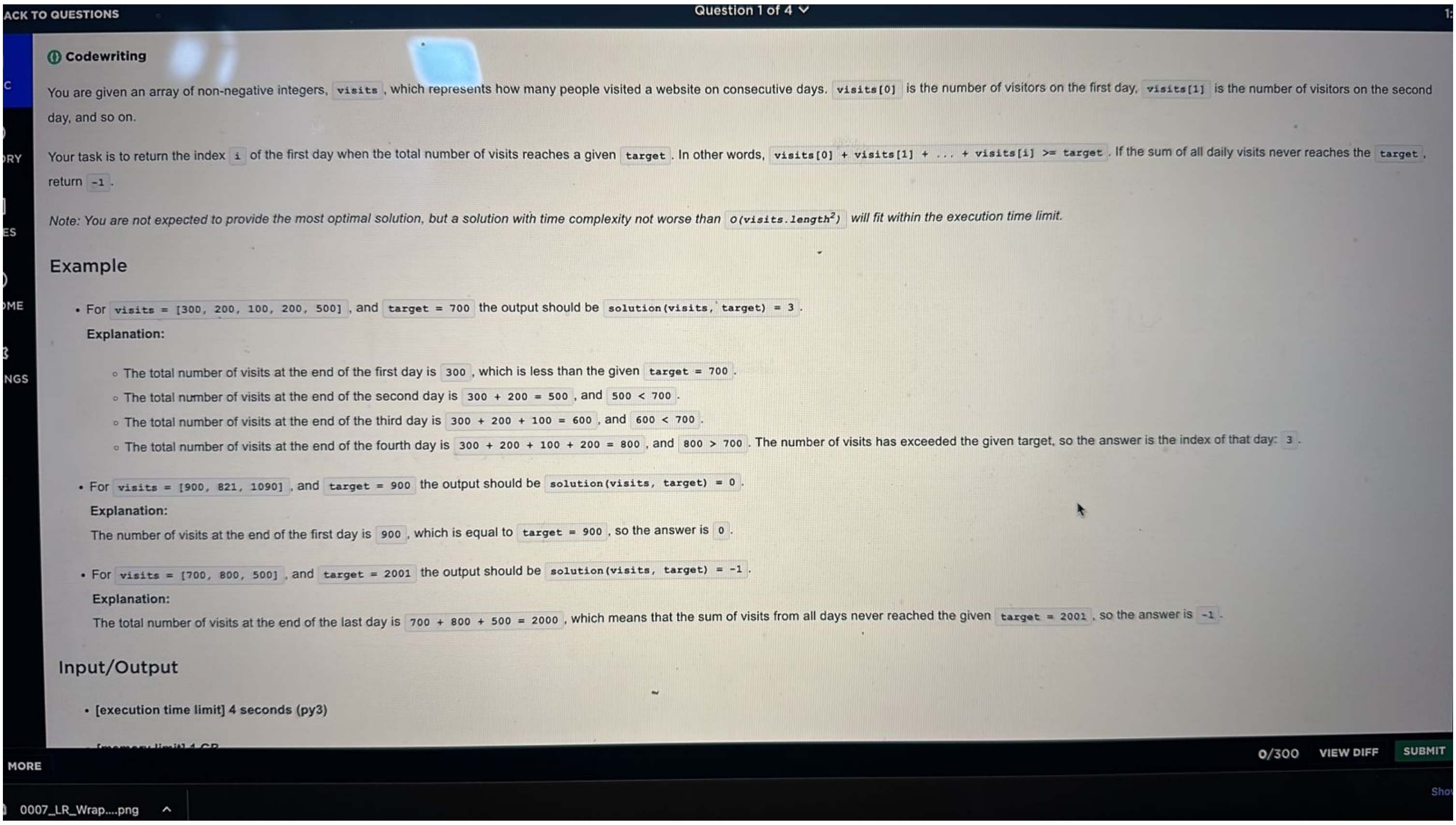
- The word `"abc"` does NOT contain any substrings that consist of the same letter repeated `3` times. Therefore, it is NOT *interesting*.
- The word `"abaaaccct"` contains two substrings satisfying the criteria (`"aaa"` and `"ccc"`), so it is *interesting*.
- The word `"abaaaa"` contains two substrings, `"aaa"` that consists of the letter `a` repeated `3` times. However, the first substring (`ab[aaa]a`) is followed by the letter `a`, and the second substring (`aba[aaa]`) is



- The word `"all"` contains a substring `"ll"` consisting of the letter `l` repeated `2` times. The letter before this substring (`a`) is different from `l` , and there are no letters after it. Therefore, the word `"all"` is *interesting*.
- The word `"cook"` contains a substring `"oo"` consisting of the letter `o` repeated `2` times. The letters both before (`c`) and after (`k`) this substring are different from `o` . Therefore, the word `"cook"` is *interesting*.
- The word `"llama"` contains a substring `"ll"` consisting of the letter `l` repeated `2` times. The letter after this substring (`a`) is different from `l` , and there are no letters before it. Therefore, the word `"llama"` is *interesting*.

Both words in `words` are *interesting*, so the answer is `3` .

- For `words = ["abc", "abaaaccct", "abaaaa", "aaaaba"]` and `n = 3` , the



Codewriting

You are given an array of non-negative integers, `visits`, which represents how many people visited a website on consecutive days. `visits[0]` is the number of visitors on the first day, `visits[1]` is the number of visitors on the second day, and so on.

Your task is to return the index `i` of the first day when the total number of visits reaches a given `target`. In other words, `visits[0] + visits[1] + ... + visits[i] >= target`. If the sum of all daily visits never reaches the `target`, return `-1`.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(\texttt{visits.length}^2)$ will fit within the execution time limit.

Example

- For `visits = [300, 200, 100, 200, 500]`, and `target = 700` the output should be `solution(visits, target) = 3`.

  Explanation:

  - The total number of visits at the end of the first day is `300`, which is less than the given `target = 700`.
  - The total number of visits at the end of the second day is `300 + 200 = 500`, and `500 < 700`.
  - The total number of visits at the end of the third day is `300 + 200 + 100 = 600`, and `600 < 700`.
  - The total number of visits at the end of the fourth day is `300 + 200 + 100 + 200 = 800`, and `800 > 700`. The number of visits has exceeded the given target, so the answer is the index of that day: `3`.

- For `visits = [900, 821, 1090]`, and `target = 900` the output should be `solution(visits, target) = 0`.

  Explanation:

  The number of visits at the end of the first day is `900`, which is equal to `target = 900`, so the answer is `0`.

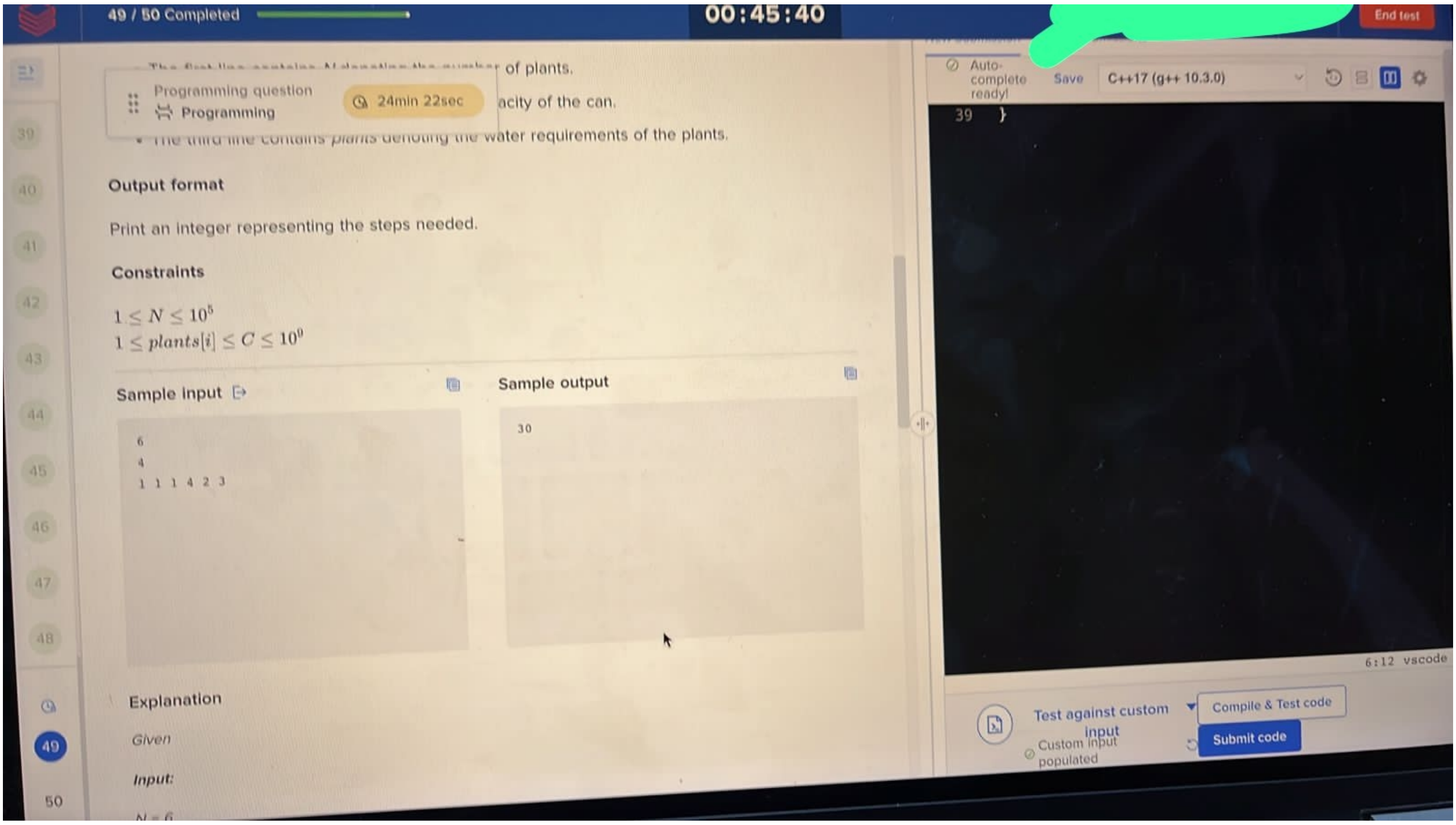
- For `visits = [700, 800, 500]`, and `target = 2001` the output should be `solution(visits, target) = -1`.

  Explanation:

  The total number of visits at the end of the last day is `700 + 800 + 500 = 2000`, which means that the sum of visits from all days never reached the given `target = 2001`, so the answer is `-1`.

Input/Output

- **[execution time limit] 4 seconds (py3)**
- **[memory limit] 1 GB**



- The first line contains N denoting the number of plants.
- …acity of the can.
- The third line contains *plants* denoting the water requirements of the plants.

Output format

Print an integer representing the steps needed.

Constraints

$1 \le N \le 10^5$
$1 \le plants[i] \le C \le 10^9$

Sample input

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

Sample output

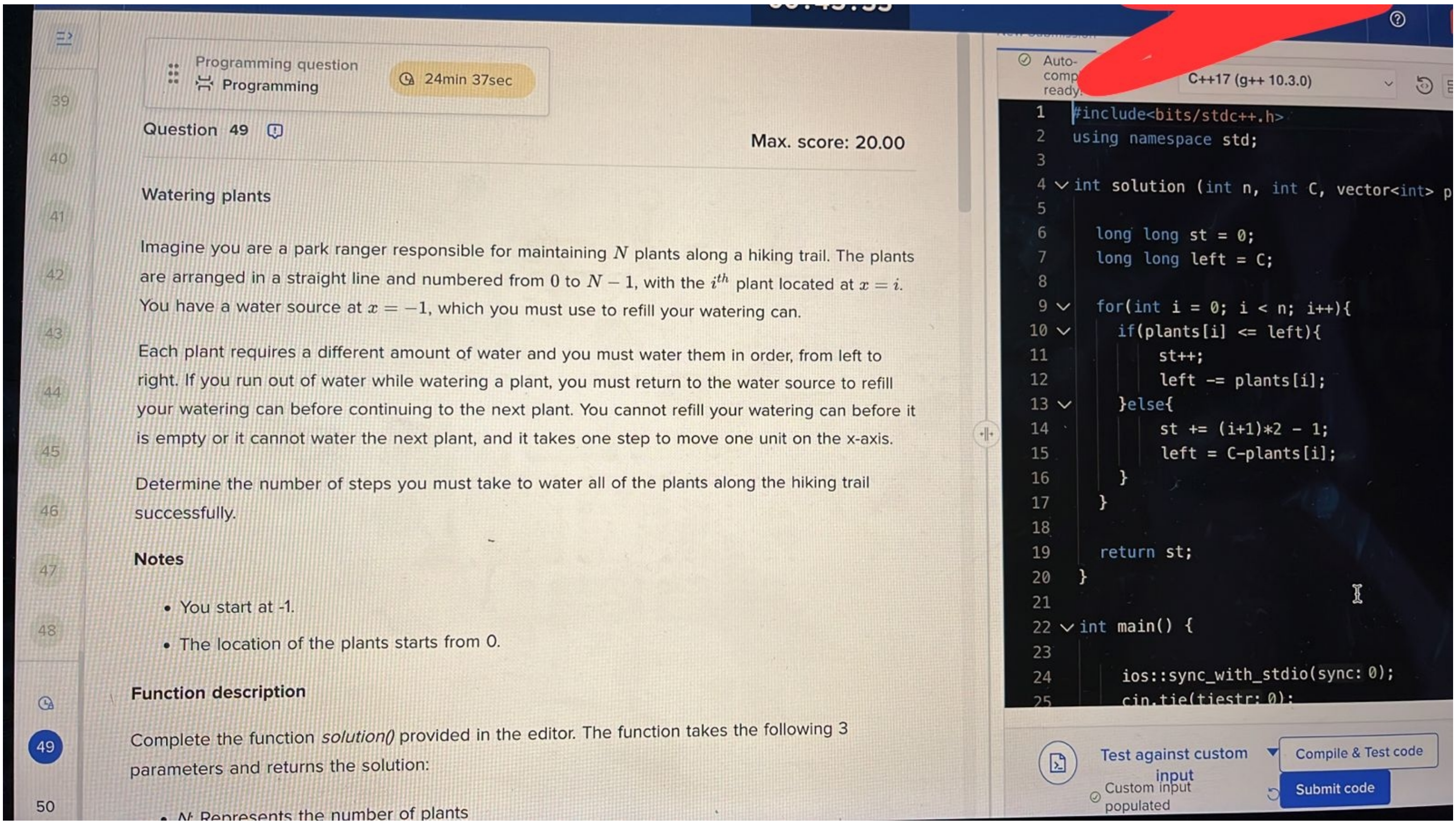
```
30
```

Explanation

Given

Input:

```
39  }
```



[redacted]

Question 49

Max. score: 20.00

Watering plants

Imagine you are a park ranger responsible for maintaining N plants along a hiking trail. The plants are arranged in a straight line and numbered from 0 to $N-1$, with the i^{th} plant located at $x = i$. You have a water source at $x = -1$, which you must use to refill your watering can.

Each plant requires a different amount of water and you must water them in order, from left to right. If you run out of water while watering a plant, you must return to the water source to refill your watering can before continuing to the next plant. You cannot refill your watering can before it is empty or it cannot water the next plant, and it takes one step to move one unit on the x-axis.

Determine the number of steps you must take to water all of the plants along the hiking trail successfully.

Notes

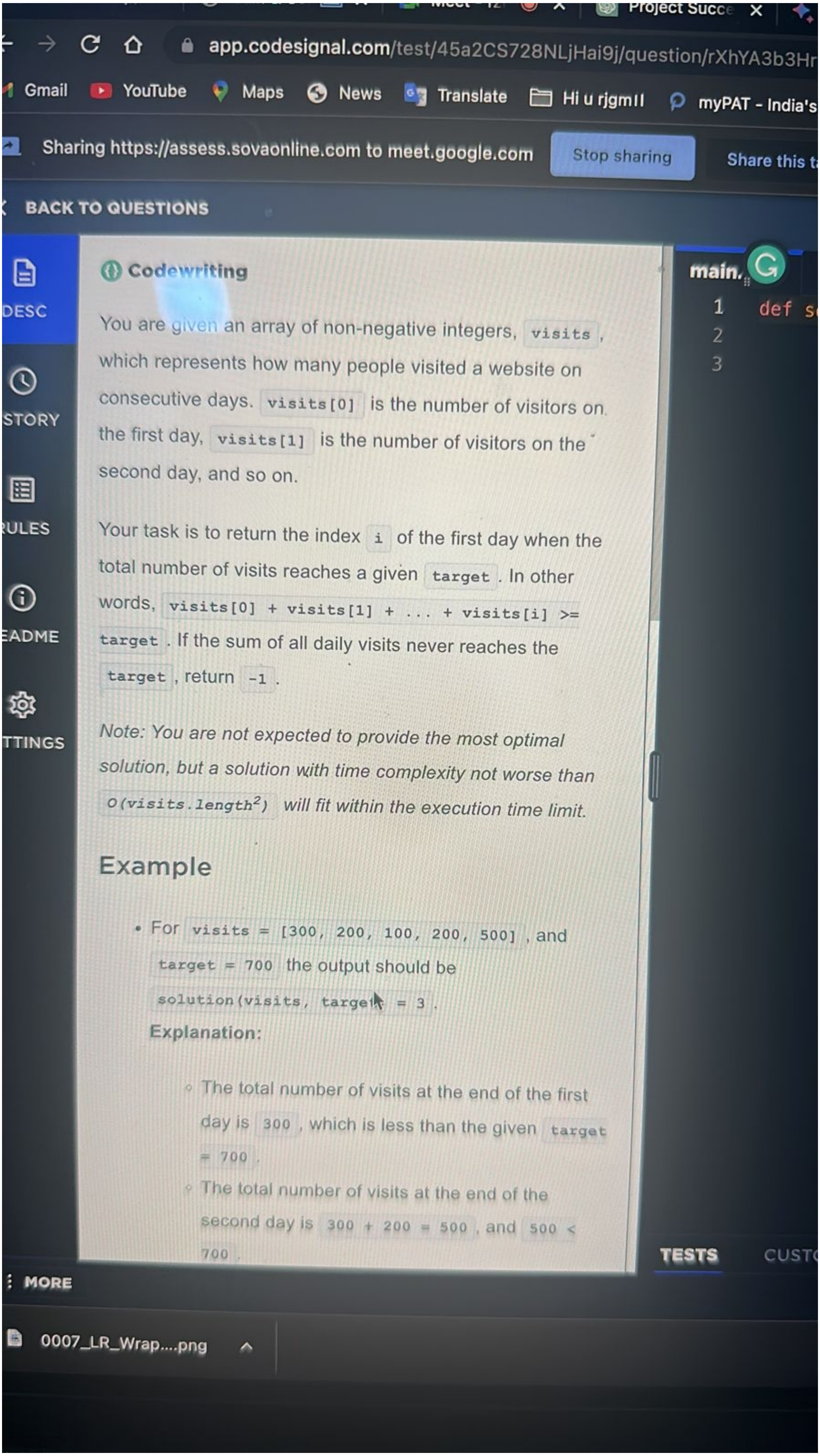
- You start at -1.
- The location of the plants starts from 0.

Function description

Complete the function *solution()* provided in the editor. The function takes the following 3 parameters and returns the solution:

- N: Represents the number of plants

```
#include<bits/stdc++.h>
using namespace std;

int solution (int n, int C, vector<int> p

    long long st = 0;
    long long left = C;

    for(int i = 0; i < n; i++){
        if(plants[i] <= left){
            st++;
            left -= plants[i];
        }else{
            st += (i+1)*2 - 1;
            left = C-plants[i];
        }
    }

    return st;
}

int main() {

    ios::sync_with_stdio(sync: 0);
    cin.tie(tiestr: 0);
```



```
def s
```

Codewriting

You are given an array of non-negative integers, `visits`, which represents how many people visited a website on consecutive days. `visits[0]` is the number of visitors on the first day, `visits[1]` is the number of visitors on the second day, and so on.

Your task is to return the index `i` of the first day when the total number of visits reaches a given `target`. In other words, `visits[0] + visits[1] + ... + visits[i] >= target`. If the sum of all daily visits never reaches the `target`, return `-1`.

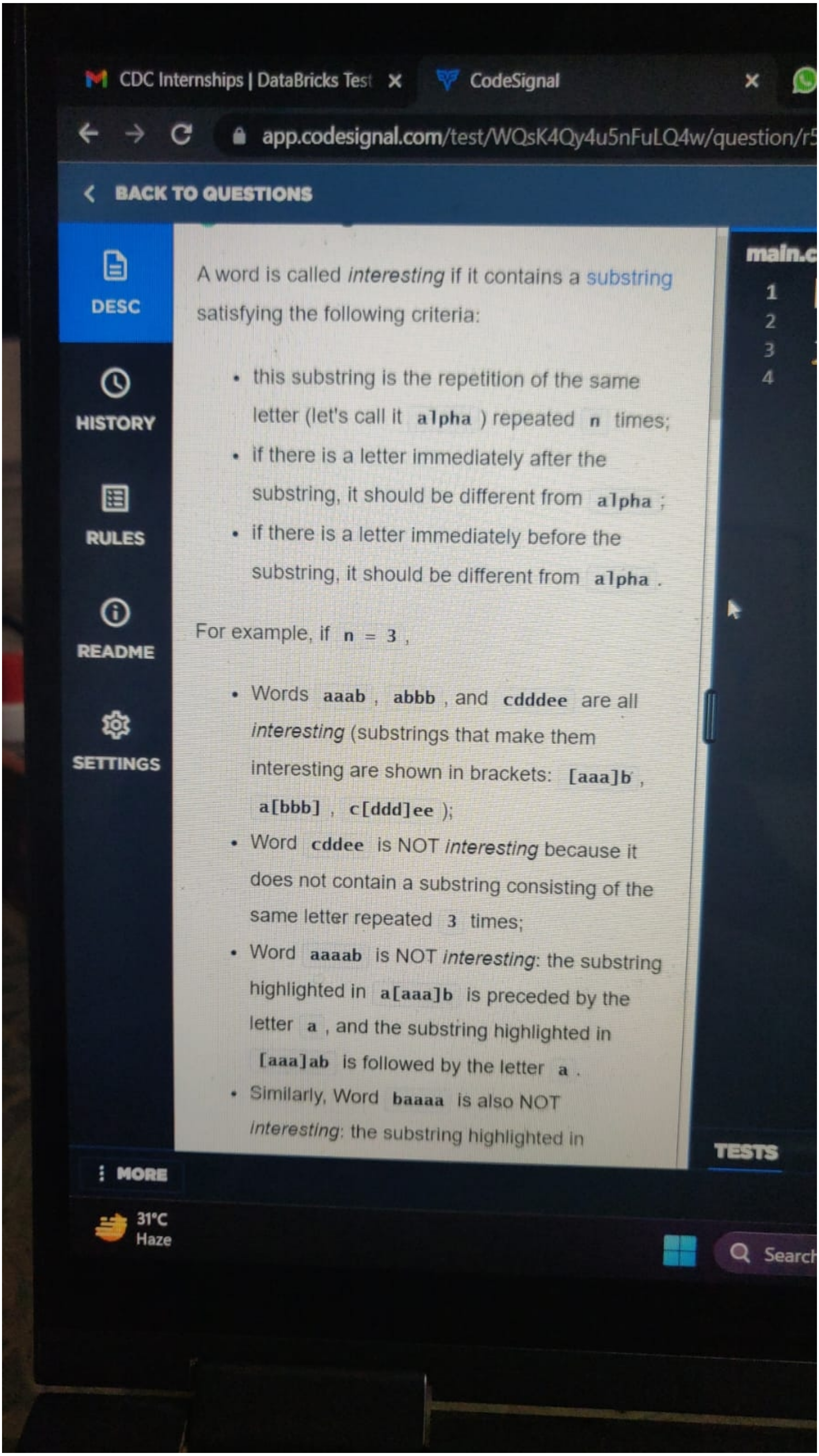
Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than `O(visits.length^2)` *will fit within the execution time limit.*

Example

- For `visits = [300, 200, 100, 200, 500]`, and `target = 700` the output should be `solution(visits, target) = 3`.

  Explanation:

  - The total number of visits at the end of the first day is `300`, which is less than the given `target = 700`.
  - The total number of visits at the end of the second day is `300 + 200 = 500`, and `500 < 700`.

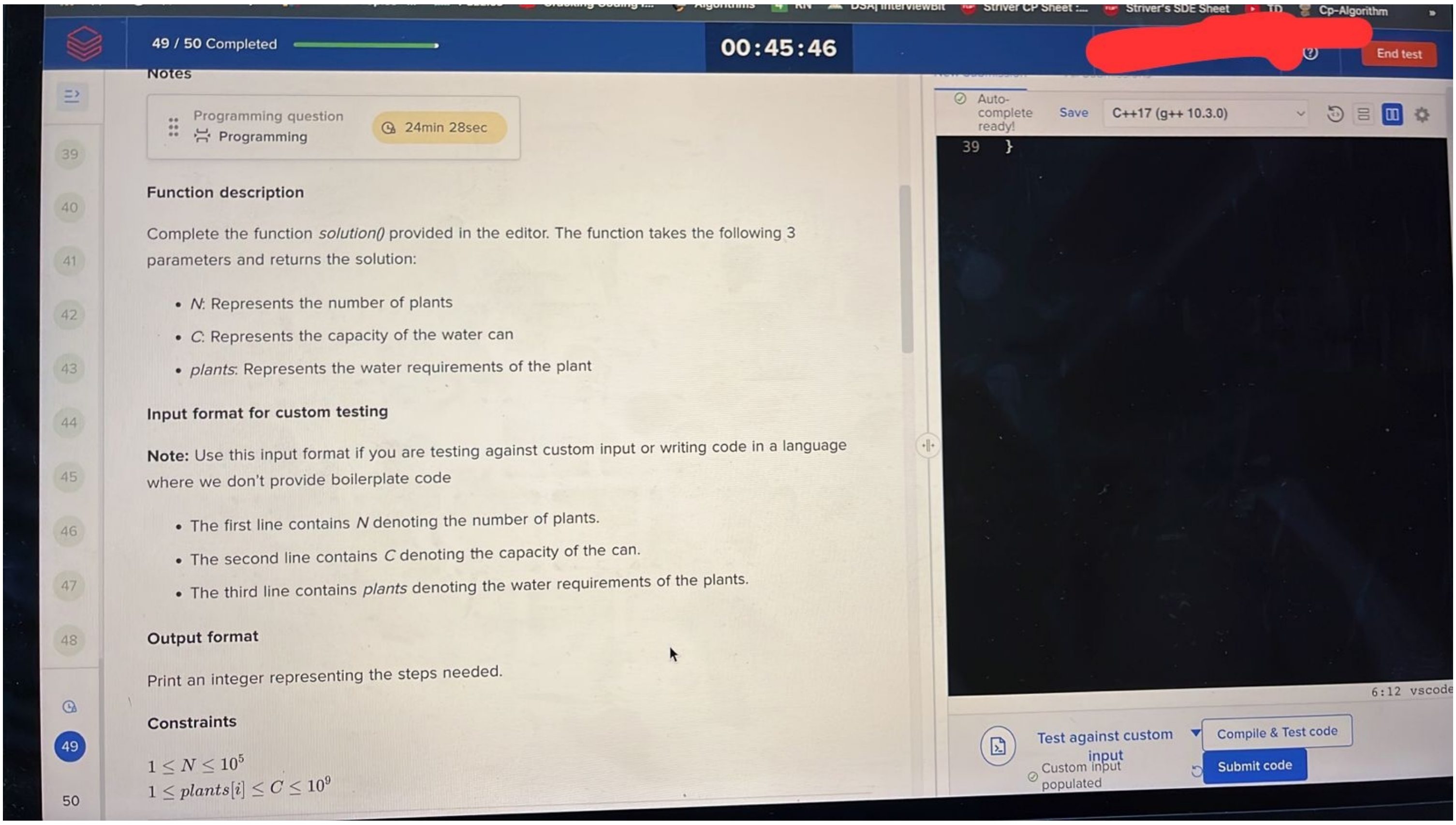

CDC Internships | DataBricks Test
CodeSignal
app.codesignal.com/test/WQsK4Qy4u5nFuLQ4w/question/r5
BACK TO QUESTIONS
DESC
HISTORY
RULES
README
SETTINGS
MORE
main.c
TESTS
31°C
Haze
Search

A word is called *interesting* if it contains a substring satisfying the following criteria:

- this substring is the repetition of the same letter (let's call it `alpha`) repeated `n` times;
- if there is a letter immediately after the substring, it should be different from `alpha` ;
- if there is a letter immediately before the substring, it should be different from `alpha` .

For example, if `n = 3` ,

- Words `aaab` , `abbb` , and `cdddee` are all *interesting* (substrings that make them interesting are shown in brackets: `[aaa]b` , `a[bbb]` , `c[ddd]ee`);
- Word `cddee` is NOT *interesting* because it does not contain a substring consisting of the same letter repeated `3` times;
- Word `aaaab` is NOT *interesting*: the substring highlighted in `a[aaa]b` is preceded by the letter `a` , and the substring highlighted in `[aaa]ab` is followed by the letter `a` .
- Similarly, Word `baaaa` is also NOT *interesting*: the substring highlighted in



Notes

Programming question
Programming
24min 28sec

Function description

Complete the function *solution()* provided in the editor. The function takes the following 3 parameters and returns the solution:

- *N*: Represents the number of plants
- *C*: Represents the capacity of the water can
- *plants*: Represents the water requirements of the plant

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don't provide boilerplate code

- The first line contains *N* denoting the number of plants.
- The second line contains *C* denoting the capacity of the can.
- The third line contains *plants* denoting the water requirements of the plants.

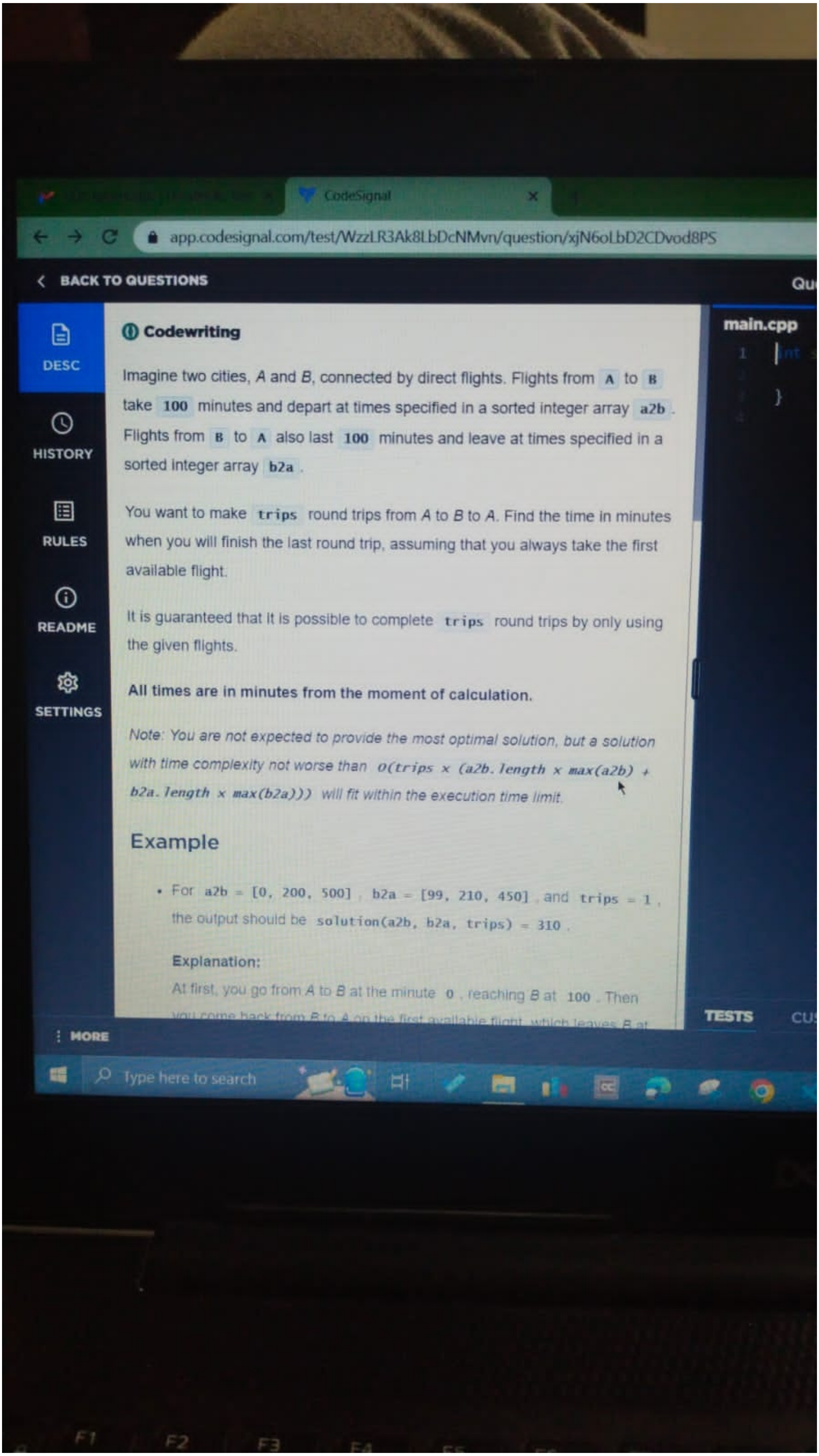
Output format

Print an integer representing the steps needed.

Constraints

$1 \leq N \leq 10^5$
$1 \leq plants[i] \leq C \leq 10^9$

```
39   }
```



Codewriting

Imagine two cities, *A* and *B*, connected by direct flights. Flights from `A` to `B` take `100` minutes and depart at times specified in a sorted integer array `a2b`. Flights from `B` to `A` also last `100` minutes and leave at times specified in a sorted integer array `b2a`.

You want to make `trips` round trips from *A* to *B* to *A*. Find the time in minutes when you will finish the last round trip, assuming that you always take the first available flight.

It is guaranteed that it is possible to complete `trips` round trips by only using the given flights.

All times are in minutes from the moment of calculation.

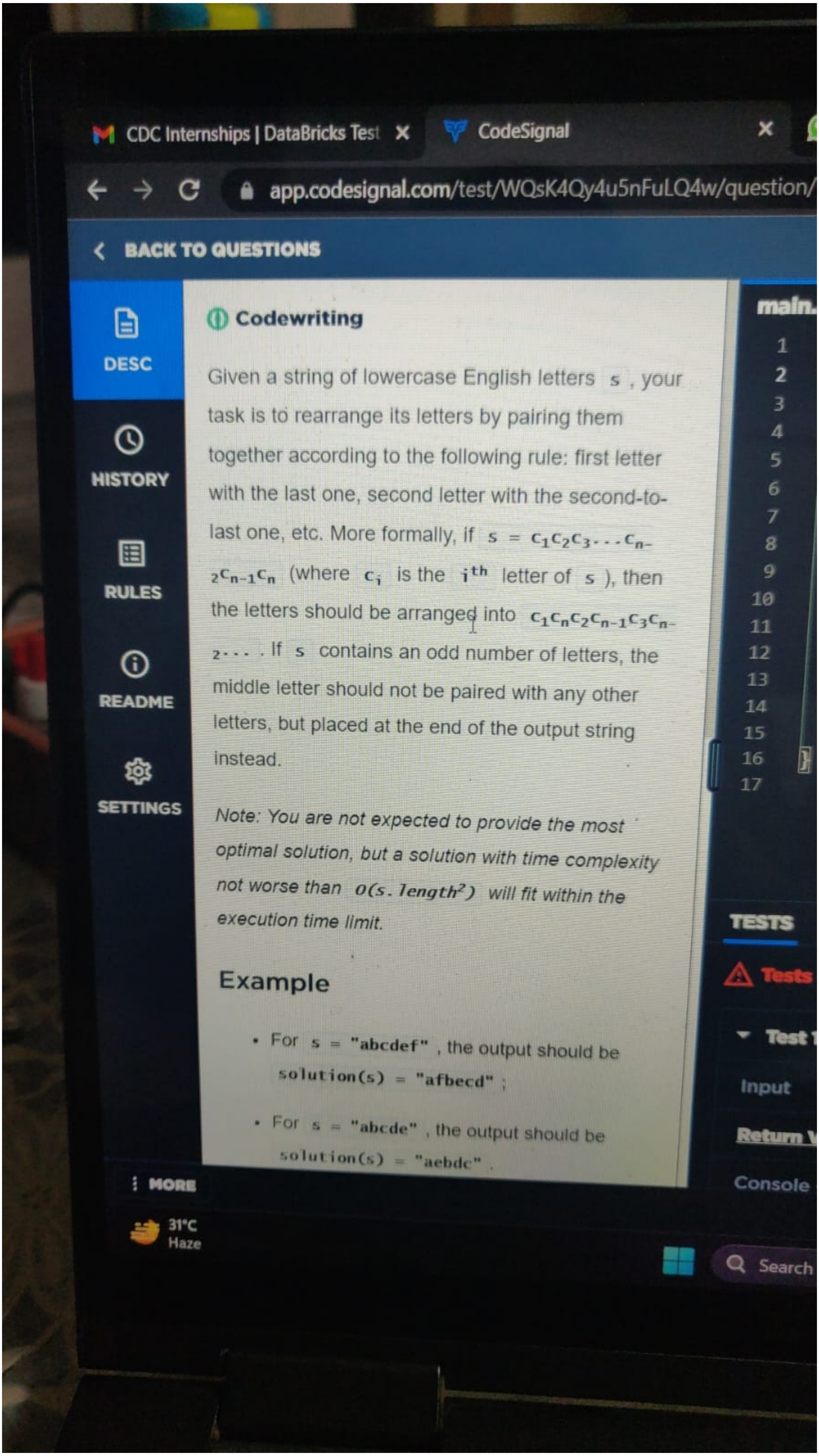
Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than `O(trips × (a2b.length × max(a2b) + b2a.length × max(b2a)))` will fit within the execution time limit.

Example

- For `a2b = [0, 200, 500]`, `b2a = [99, 210, 450]`, and `trips = 1`, the output should be `solution(a2b, b2a, trips) = 310`.

Explanation:

At first, you go from *A* to *B* at the minute `0`, reaching *B* at `100`. Then you come back from *B* to *A* on the first available flight, which leaves *B* at



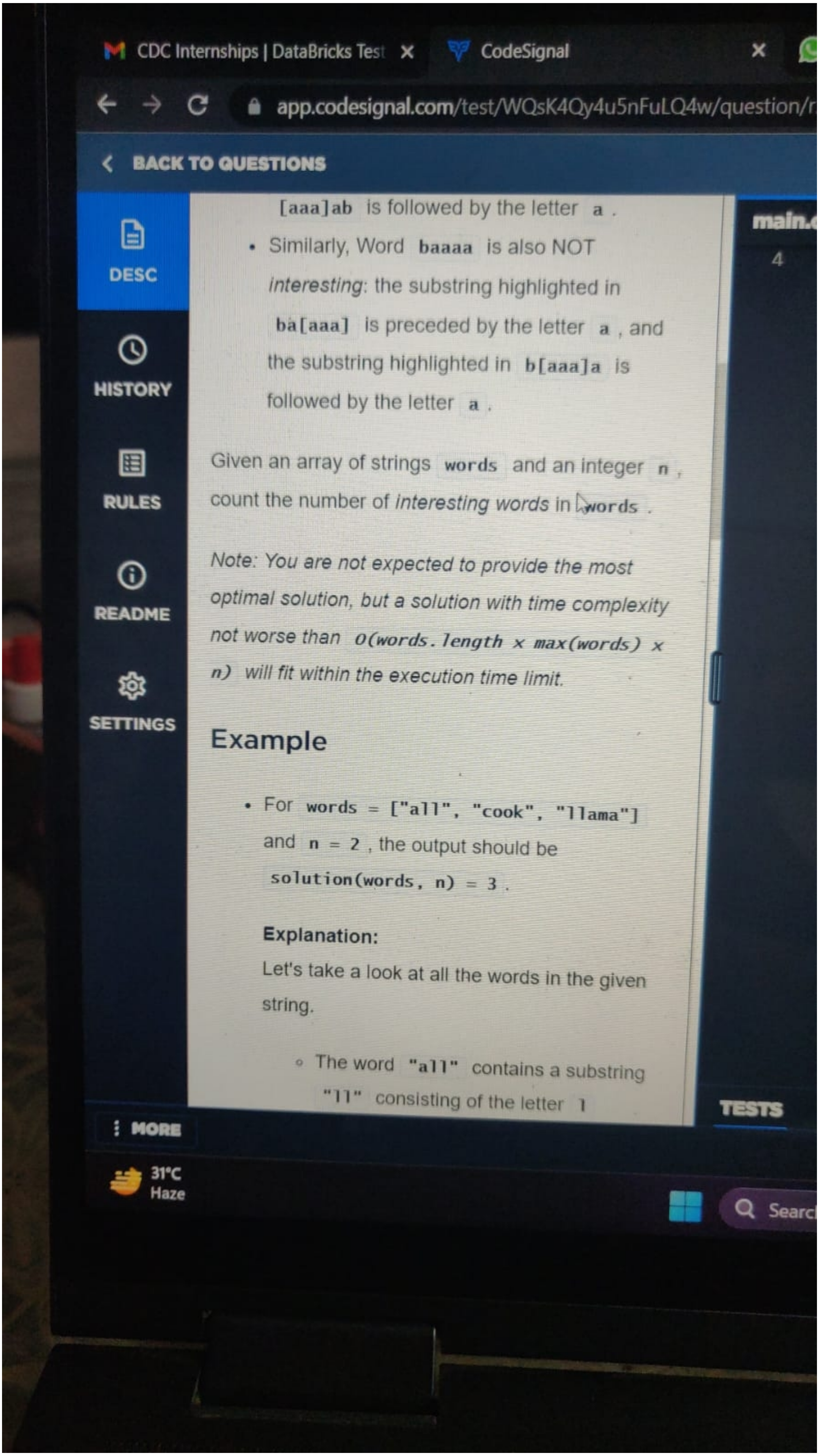
Codewriting

Given a string of lowercase English letters `s`, your task is to rearrange its letters by pairing them together according to the following rule: first letter with the last one, second letter with the second-to-last one, etc. More formally, if $s = c_1c_2c_3...c_{n-2}c_{n-1}c_n$ (where c_i is the i^{th} letter of `s`), then the letters should be arranged into $c_1c_nc_2c_{n-1}c_3c_{n-2}...$. If `s` contains an odd number of letters, the middle letter should not be paired with any other letters, but placed at the end of the output string instead.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than $O(s.length^2)$ will fit within the execution time limit.

Example

- For `s = "abcdef"`, the output should be `solution(s) = "afbecd"`;
- For `s = "abcde"`, the output should be `solution(s) = "aebdc"`.


CDC Internships | DataBricks Test
CodeSignal
app.codesignal.com/test/WQsK4Qy4u5nFuLQ4w/question/r
BACK TO QUESTIONS
DESC
HISTORY
RULES
README
SETTINGS
MORE
main.
4
TESTS
31°C
Haze
Searc

`[aaa]ab` is followed by the letter `a`.

- Similarly, Word `baaaa` is also NOT *interesting*: the substring highlighted in `ba[aaa]` is preceded by the letter `a`, and the substring highlighted in `b[aaa]a` is followed by the letter `a`.

Given an array of strings `words` and an integer `n`, count the number of *interesting* words in `words`.

Note: You are not expected to provide the most optimal solution, but a solution with time complexity not worse than `O(words.length x max(words) x n)` will fit within the execution time limit.

Example

- For `words = ["all", "cook", "llama"]` and `n = 2`, the output should be `solution(words, n) = 3`.

  Explanation:

  Let's take a look at all the words in the given string.

  - The word `"all"` contains a substring `"ll"` consisting of the letter `l`