

```
h  
1 def fibonacci(n):  
2     if n < 0:  
3         ra  
random import  
range(stop) funct  
range(start, stop[, step]) funct  
raw_input([prompt]) funct  
raw_input keyw  
range keyw  
raise keyw
```

Please rate your test experience

Clarity of questions	★ ★ ★ ★ ★
Usability of Test Interface	★ ★ ★ ★ ★
Usability of Code Editor	★ ★ ★ ★ ★
Fairness of skill assessment	★ ★ ★ ★ ★

Good

Logout of the current session

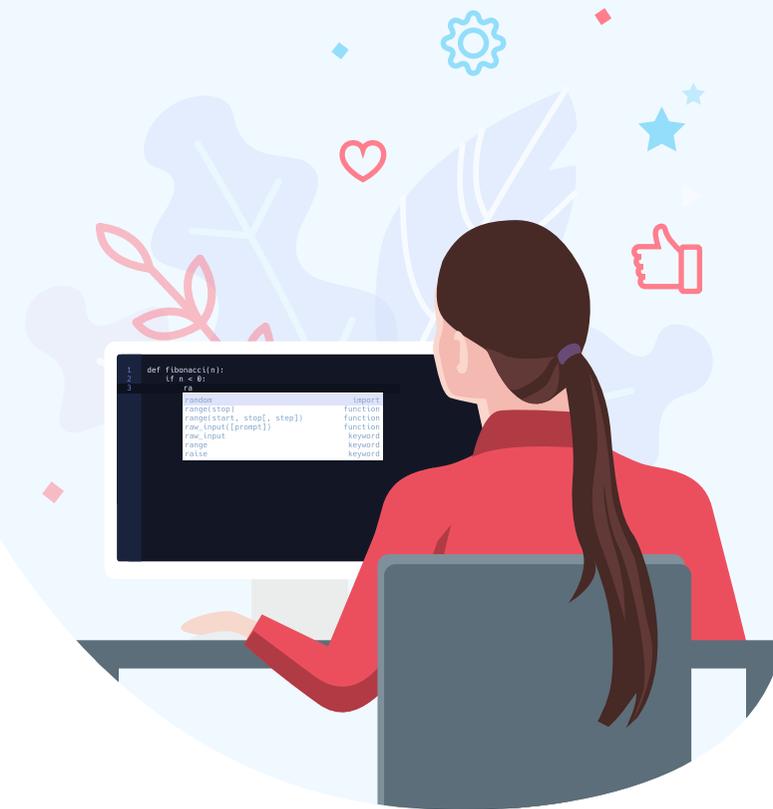
[Submit](#)

[Skip](#)

# How to improve **candidate experience** using HackerEarth

## Enhancing candidate experience using developer assessments

When an organization evaluates an assessment platform for their screening process, they also look at how optimal the candidate experience is going to be. A negative candidate experience can not only lead to organizations missing out on hiring potential candidates, but also incur a significant loss of trust and revenue. On the contrary, a positive experience can benefit the organization in building a strong talent pipeline.

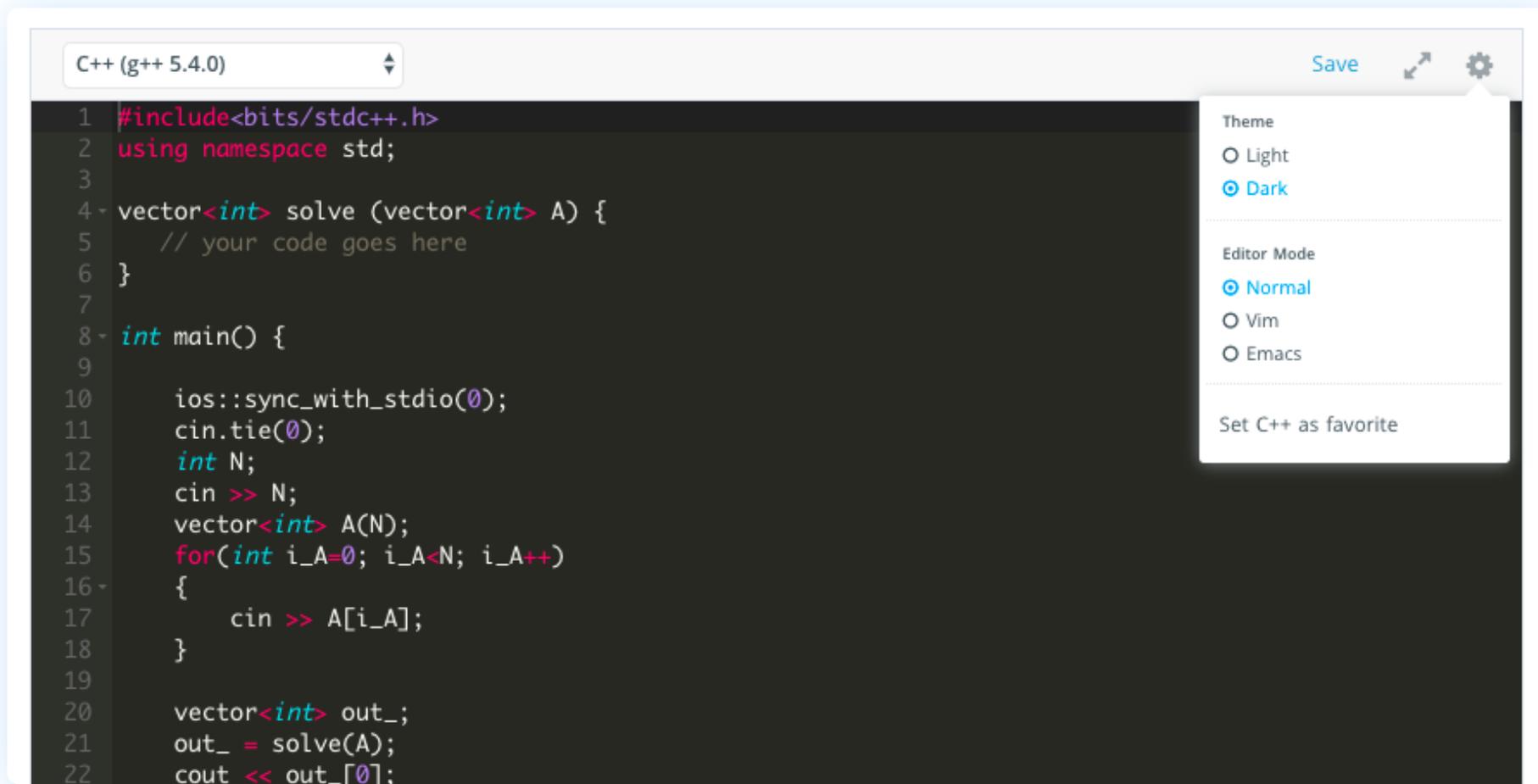


Ensuring a good candidate experience is a crucial part of an organization's talent acquisition strategy. Thus, they need to be watchful of the assessment platform they use in their hiring process. Here's how you can use HackerEarth to provide an ideal test experience for candidates.

## Bring out the best in your candidates

### Allow candidates to code more efficiently using online IDEs

The HackerEarth platform offers multiple IDE options (such as VIM and Emacs) that are available for a smooth user interface. These options enable candidates to write, debug, and compile their code more effectively. Also, based on their preference, candidates can choose to toggle between editor modes, code in a light or dark theme, and even set a programming language as their favorite.



The image shows a screenshot of an online IDE interface. The top bar displays "C++ (g++ 5.4.0)" on the left and "Save" with a share icon and a settings gear icon on the right. The main editor area contains the following C++ code:

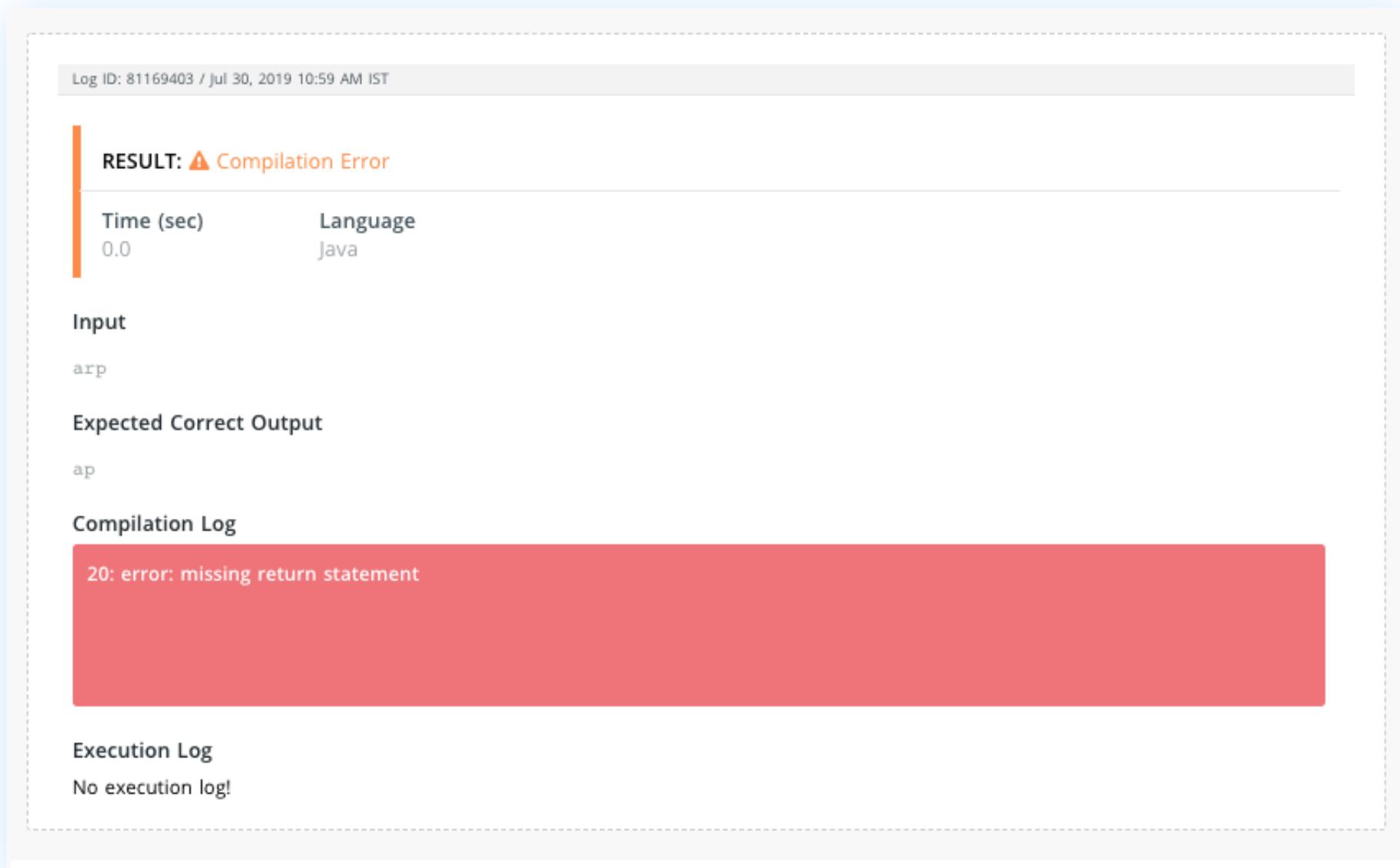
```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 vector<int> solve (vector<int> A) {
5     // your code goes here
6 }
7
8 int main() {
9
10     ios::sync_with_stdio(0);
11     cin.tie(0);
12     int N;
13     cin >> N;
14     vector<int> A(N);
15     for(int i_A=0; i_A<N; i_A++)
16     {
17         cin >> A[i_A];
18     }
19
20     vector<int> out_;
21     out_ = solve(A);
22     cout << out_[0];
```

On the right side, a settings menu is open, showing the following options:

- Theme
  - Light
  - Dark
- Editor Mode
  - Normal
  - Vim
  - Emacs
- Set C++ as favorite

## Save a candidate's time while coding

While executing their code, candidates can directly view compilation and runtime errors below the code editor interface, making it easier for candidates to fix their code before they submit it.



The screenshot displays a coding interface with a log header and several sections. The log header shows 'Log ID: 81169403 / Jul 30, 2019 10:59 AM IST'. Below this, a 'RESULT' section indicates a 'Compilation Error' with a warning icon. A table shows the execution time as 0.0 seconds for the language Java. The 'Input' section contains the text 'arp', and the 'Expected Correct Output' section contains 'ap'. The 'Compilation Log' section features a red background with the message '20: error: missing return statement'. Finally, the 'Execution Log' section shows 'No execution log!'.

Log ID: 81169403 / Jul 30, 2019 10:59 AM IST

**RESULT:** ⚠️ **Compilation Error**

Time (sec)	Language
0.0	Java

**Input**

arp

**Expected Correct Output**

ap

**Compilation Log**

20: error: missing return statement

**Execution Log**

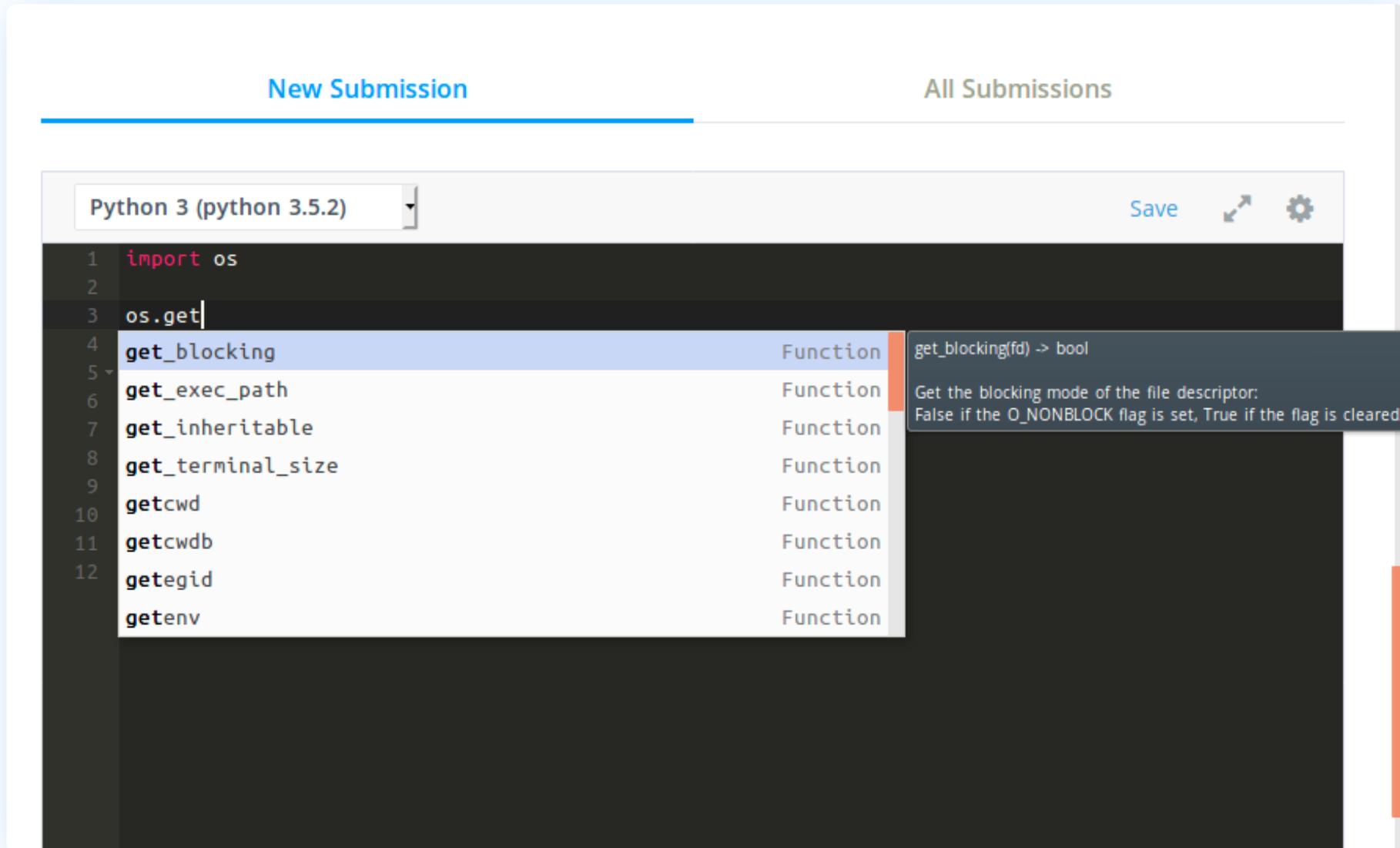
No execution log!

In addition, the pre-populated code snippets and autocomplete features in the HackerEarth platform let candidates focus on logic rather than syntax.

Sample code snippet for Java

```
1 import java.io.*;
2 import java.util.*;
3
4
5 public class TestClass {
6     public static void main(String[] args) throws IOException {
7         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
8         PrintWriter wr = new PrintWriter(System.out);
9         String S = br.readLine();
10
11         String out_ = solve(S);
12         System.out.println(out_);
13
14         wr.close();
15         br.close();
16     }
17     static String solve(String S){
18         // Write your code here
19     }
20 }
21 }
```

## Using autocomplete feature in HackerEarth



The screenshot shows the HackerEarth code editor interface. At the top, there are two tabs: "New Submission" (active) and "All Submissions". Below the tabs is a header bar with a dropdown menu set to "Python 3 (python 3.5.2)", a "Save" button, and icons for a share link and settings. The main area is a code editor with a dark background. The code on the screen is:

```
1 import os
2
3 os.get|
```

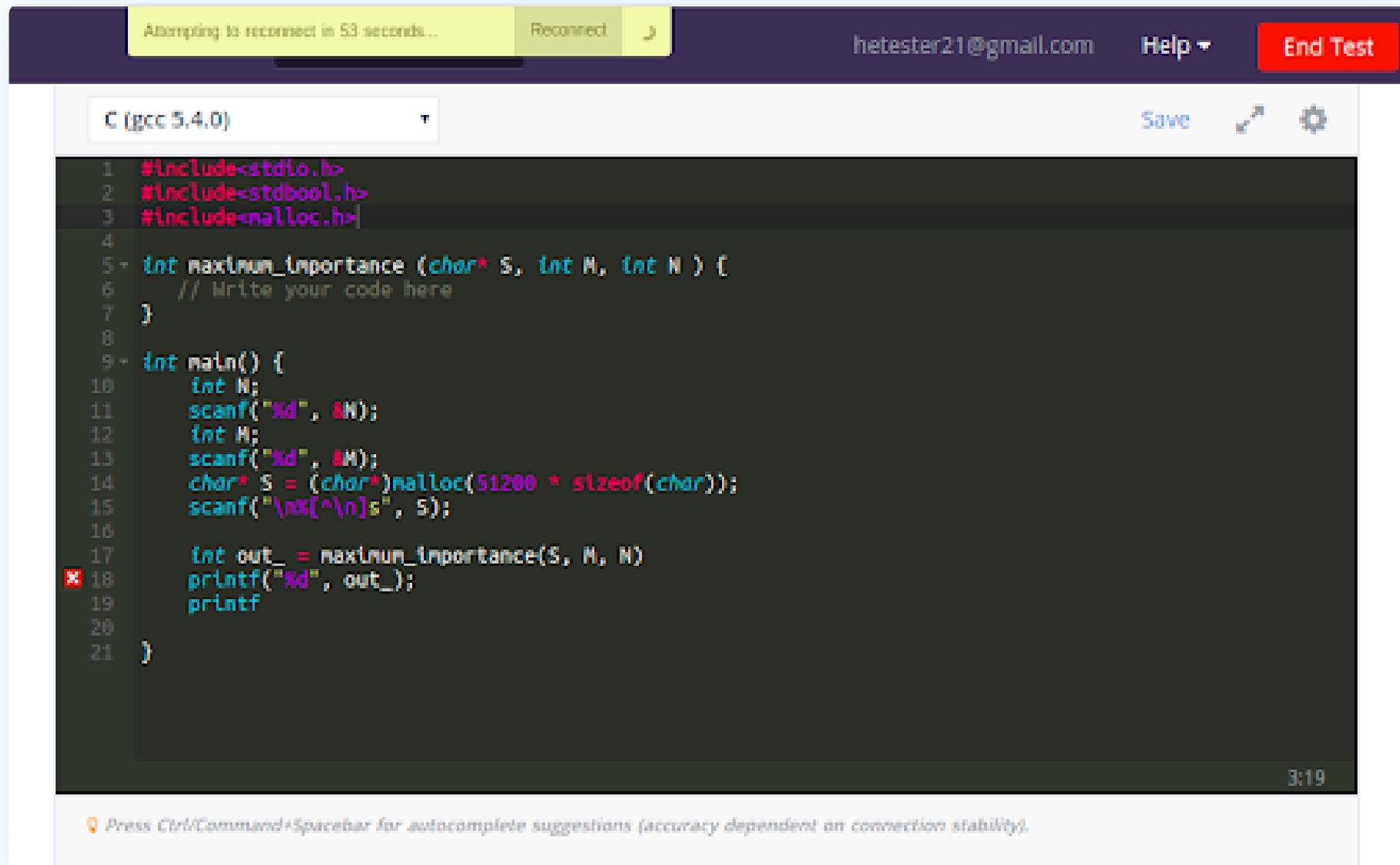
An autocomplete dropdown menu is open below the cursor, listing several methods from the `os` module. Each item includes the method name, its type (all are "Function"), and a tooltip for the selected item, `get_blocking`. The tooltip text is:

```
get_blocking(fd) -> bool
Get the blocking mode of the file descriptor:
False if the O_NONBLOCK flag is set, True if the flag is cleared.
```

Method Name	Type
<code>get_blocking</code>	Function
<code>get_exec_path</code>	Function
<code>get_inheritable</code>	Function
<code>get_terminal_size</code>	Function
<code>getcwd</code>	Function
<code>getcwdb</code>	Function
<code>getegid</code>	Function
<code>getenv</code>	Function

## Provide a smooth test experience using proactive alerts

If there's an error in the test environment, candidates receive alerts with appropriate error messages. These can include issues related to network failure, server error, errors in loading files, and more.



The screenshot shows a code editor interface with a dark theme. At the top, a yellow banner displays the message "Attempting to reconnect in 53 seconds..." with a "Reconnect" button. The user's email "hetester21@gmail.com" and a "Help" menu are visible on the right, along with a red "End Test" button. The editor title is "C (gcc 5.4.0)" and includes "Save" and "Settings" icons. The code is as follows:

```
1 #include<stdio.h>
2 #include<stdbool.h>
3 #include<malloc.h>
4
5 int maximum_importance (char* S, int M, int N) {
6     // Write your code here
7 }
8
9 int main() {
10     int M;
11     scanf("%d", &M);
12     int N;
13     scanf("%d", &N);
14     char* S = (char*)malloc(51200 * sizeof(char));
15     scanf("%s[%s]", S);
16
17     int out_ = maximum_importance(S, M, N)
18     printf("%d", out_);
19     printf
20
21 }
```

A red 'x' icon is next to line 18, indicating a compilation error. The bottom right corner shows a timer at 3:19. A footer note reads: "Press Ctrl/Command+Spacebar for autocomplete suggestions (accuracy dependent on connection stability)."

## Make assessments more user-friendly

### Code in the language of their choice

Based on admin settings, candidates can choose from 38 different programming languages to answer programming questions

#### Allowed Languages

Select all

- |   |  |   |
|---|--|---|
| <input checked="" type="checkbox"/> Bash                | <input checked="" type="checkbox"/> C            | <input checked="" type="checkbox"/> C++               |
| <input checked="" type="checkbox"/> C++14               | <input checked="" type="checkbox"/> Clojure      | <input checked="" type="checkbox"/> C#                |
| <input checked="" type="checkbox"/> D                   | <input checked="" type="checkbox"/> Erlang       | <input checked="" type="checkbox"/> F#                |
| <input checked="" type="checkbox"/> Go                  | <input checked="" type="checkbox"/> Groovy       | <input checked="" type="checkbox"/> Haskell           |
| <input checked="" type="checkbox"/> Java                | <input checked="" type="checkbox"/> Java 8       | <input checked="" type="checkbox"/> JavaScript(Rhino) |
| <input checked="" type="checkbox"/> JavaScript(Node.js) | <input checked="" type="checkbox"/> Julia        | <input checked="" type="checkbox"/> Kotlin            |
| <input checked="" type="checkbox"/> Lisp                | <input checked="" type="checkbox"/> Lisp (SBCL)  | <input checked="" type="checkbox"/> Lua               |
| <input checked="" type="checkbox"/> Objective-C         | <input checked="" type="checkbox"/> OCaml        | <input checked="" type="checkbox"/> Octave            |
| <input checked="" type="checkbox"/> Pascal              | <input checked="" type="checkbox"/> Perl         | <input checked="" type="checkbox"/> PHP               |
| <input checked="" type="checkbox"/> Python              | <input checked="" type="checkbox"/> Python 3     | <input checked="" type="checkbox"/> R(RScript)        |
| <input checked="" type="checkbox"/> Racket              | <input checked="" type="checkbox"/> Ruby         | <input checked="" type="checkbox"/> Rust              |
| <input checked="" type="checkbox"/> Scala               | <input checked="" type="checkbox"/> Swift        | <input checked="" type="checkbox"/> Swift-4.1         |
| <input checked="" type="checkbox"/> TypeScript          | <input checked="" type="checkbox"/> Visual Basic |   |

### Take assessments in different languages

The HackerEarth platform allows candidates to take an assessment in five other native languages apart from English

#### Language Settings

##### Test Language

Set the language in which the test has to be opened initially

English

##### Allowed Languages

Set multiple languages in which a candidate can toggle and view the test

- English
- Japanese
- Chinese
- French
- Portuguese
- Russian

Save Language Settings

# Provide room for improvement

## Prepare for assessments by taking practice tests

Candidates can practice tests and get familiar with the test environment. This will help them in honing their skills before attempting an actual assessment.

The screenshot shows the HackerEarth Practice Test interface. At the top, there is a dark blue header with the HackerEarth logo, the text "Practice Test" with a "Preview" button, a timer showing "1:30:00 left", the user email "pritika@hackerearth.com", and a language dropdown set to "English". Below the header is a progress bar with four steps: "1 Enter the test" (active), "2 Enter your details", "3 Read Instructions", and "4 Solve Questions".

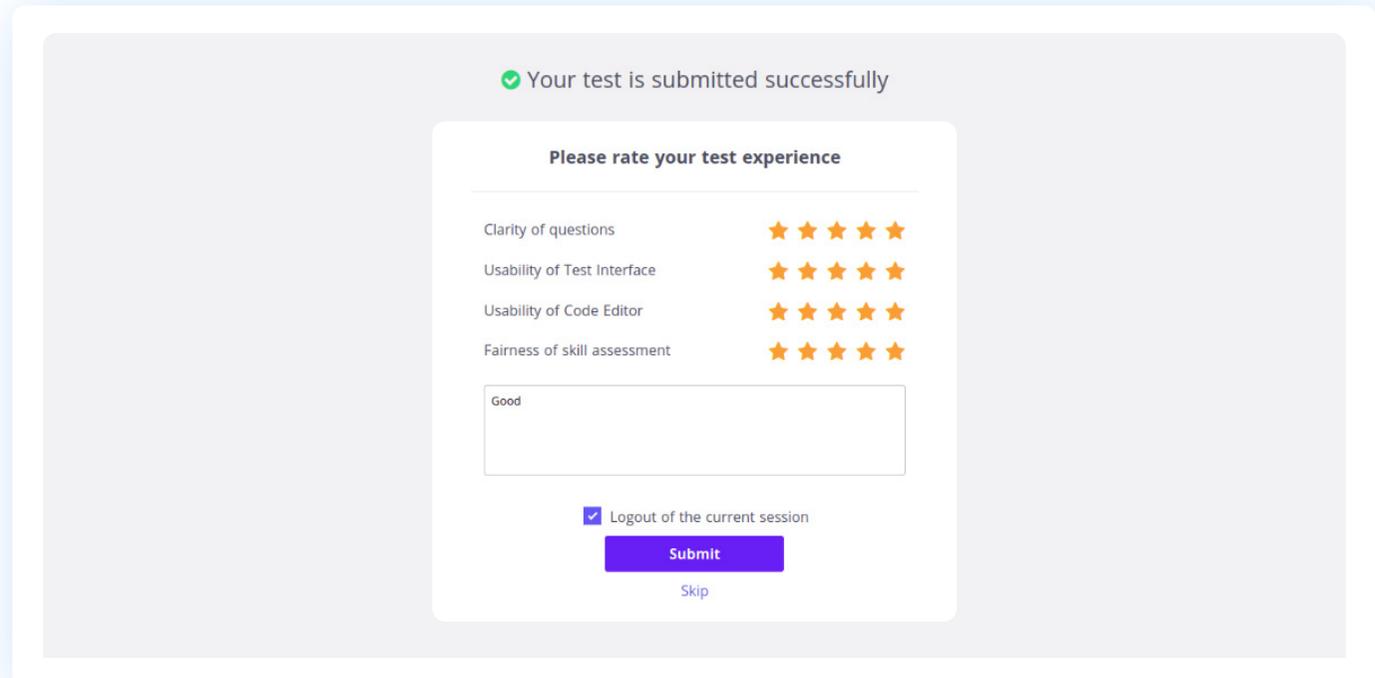
The main content area is divided into two columns. The left column contains the test title "Practice Test", the creator "By HackerEarth", and the duration "Duration: 1 hrs 30 mins". The right column features a green checkmark indicating "System is compatible for taking this test", a checkbox for agreeing to the "Privacy Policy and Terms of Use", and a large green "Enter Test" button.

Below the main content, there are three sections: "About this Test" (describing the 1hr 30 min duration and purpose), "General instructions" (listing rules like test duration, email ID, and submit button), and "Recommended before you start" (providing contact information for support at support@hackerearth.com). A "Help & Support" section is also present. A question mark icon is visible in the bottom right corner.

<https://www.hackerearth.com/challenges/test/practice-test/start-test/>

## Make your assessments more relevant with candidate feedback

After completing an assessment, candidates can give detailed feedback on the HackerEarth platform, informing recruiters about their experience and if there's any scope of improvement.



✔ Your test is submitted successfully

Please rate your test experience

Clarity of questions	★★★★★
Usability of Test Interface	★★★★★
Usability of Code Editor	★★★★★
Fairness of skill assessment	★★★★★

Good

Logout of the current session

Try taking an assessment on HackerEarth and let us know how it works for you. If you need any help using the HackerEarth platform, write back to us at [support@hackerearth.com](mailto:support@hackerearth.com). If you're new to HackerEarth and want to create accurate skill-based developer assessments, [sign up for a 14-day free trial](#).