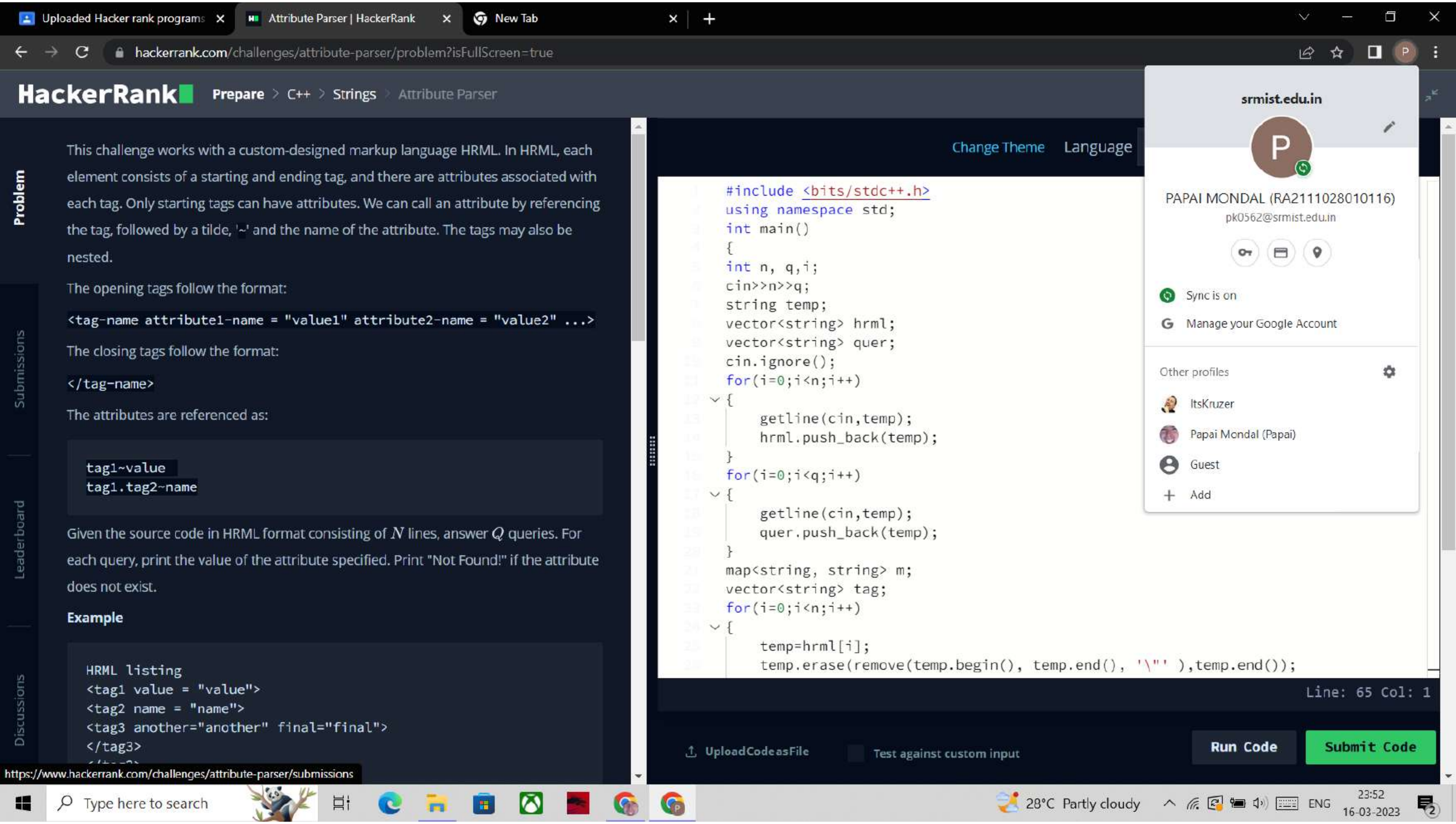
1. Attribute Parser


Uploaded Hacker rank programs
Attribute Parser | HackerRank
New Tab
hackerrank.com/challenges/attribute-parser/problem?isFullScreen=true
HackerRank
Prepare > C++ > Strings > Attribute Parser
Problem
Submissions
Leaderboard
Discussions
Change Theme
Language
srmist.edu.in
PAPAI MONDAL (RA2111028010116)
pk0562@srmist.edu.in
Sync is on
Manage your Google Account
Other profiles
ItsKruzer
Papai Mondal (Papai)
Guest
Add
Line: 65 Col: 1
UploadCodeasFile
Test against custom input
Run Code
Submit Code
https://www.hackerrank.com/challenges/attribute-parser/submissions
Type here to search
28°C Partly cloudy
ENG
23:52
16-03-2023

This challenge works with a custom-designed markup language HRML. In HRML, each element consists of a starting and ending tag, and there are attributes associated with each tag. Only starting tags can have attributes. We can call an attribute by referencing the tag, followed by a tilde, '~' and the name of the attribute. The tags may also be nested.

The opening tags follow the format:

`<tag-name attribute1-name = "value1" attribute2-name = "value2" ...>`

The closing tags follow the format:

`</tag-name>`

The attributes are referenced as:

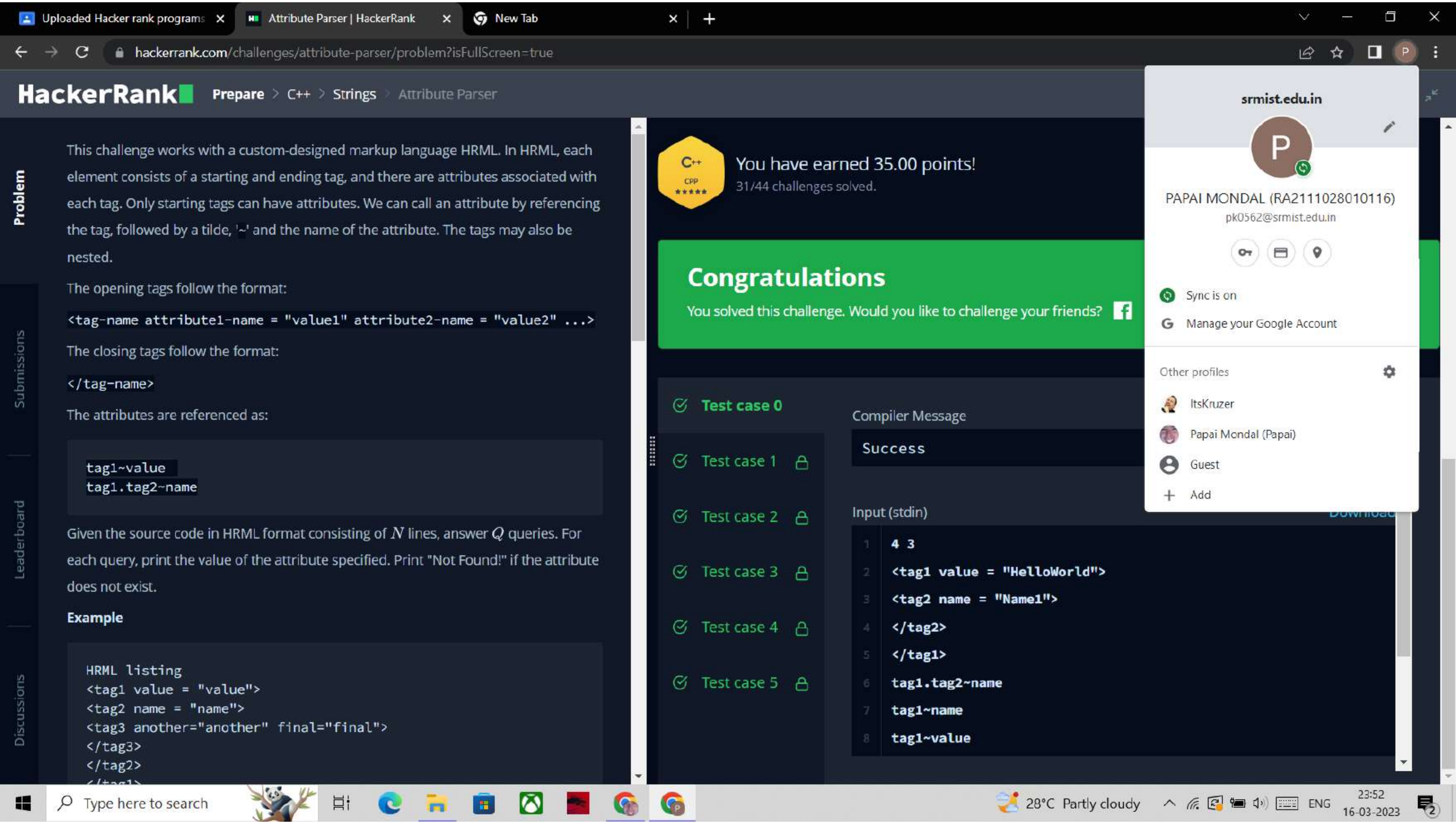
```
tag1~value
tag1.tag2~name
```

Given the source code in HRML format consisting of N lines, answer Q queries. For each query, print the value of the attribute specified. Print "Not Found!" if the attribute does not exist.

Example

```
HRML listing
<tag1 value = "value">
<tag2 name = "name">
<tag3 another="another" final="final">
</tag3>
```

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
int n, q,i;
cin>>n>>q;
string temp;
vector<string> hrml;
vector<string> quer;
cin.ignore();
for(i=0;i<n;i++)
{
    getline(cin,temp);
    hrml.push_back(temp);
}
for(i=0;i<q;i++)
{
    getline(cin,temp);
    quer.push_back(temp);
}
map<string, string> m;
vector<string> tag;
for(i=0;i<n;i++)
{
    temp=hrml[i];
    temp.erase(remove(temp.begin(), temp.end(), '\"' ),temp.end());
```



Uploaded Hacker rank programs
Attribute Parser | HackerRank
New Tab
hackerrank.com/challenges/attribute-parser/problem?isFullScreen=true
HackerRank
Prepare > C++ > Strings > Attribute Parser
Problem
Submissions
Leaderboard
Discussions

This challenge works with a custom-designed markup language HRML. In HRML, each element consists of a starting and ending tag, and there are attributes associated with each tag. Only starting tags can have attributes. We can call an attribute by referencing the tag, followed by a tilde, '~' and the name of the attribute. The tags may also be nested.

The opening tags follow the format:

```
<tag-name attribute1-name = "value1" attribute2-name = "value2" ...>
```

The closing tags follow the format:

```
</tag-name>
```

The attributes are referenced as:

```
tag1~value
tag1.tag2~name
```

Given the source code in HRML format consisting of N lines, answer Q queries. For each query, print the value of the attribute specified. Print "Not Found!" if the attribute does not exist.

Example

```
HRML listing
<tag1 value = "value">
<tag2 name = "name">
<tag3 another="another" final="final">
</tag3>
</tag2>
```

You have earned 35.00 points!
31/44 challenges solved.

Congratulations
You solved this challenge. Would you like to challenge your friends?

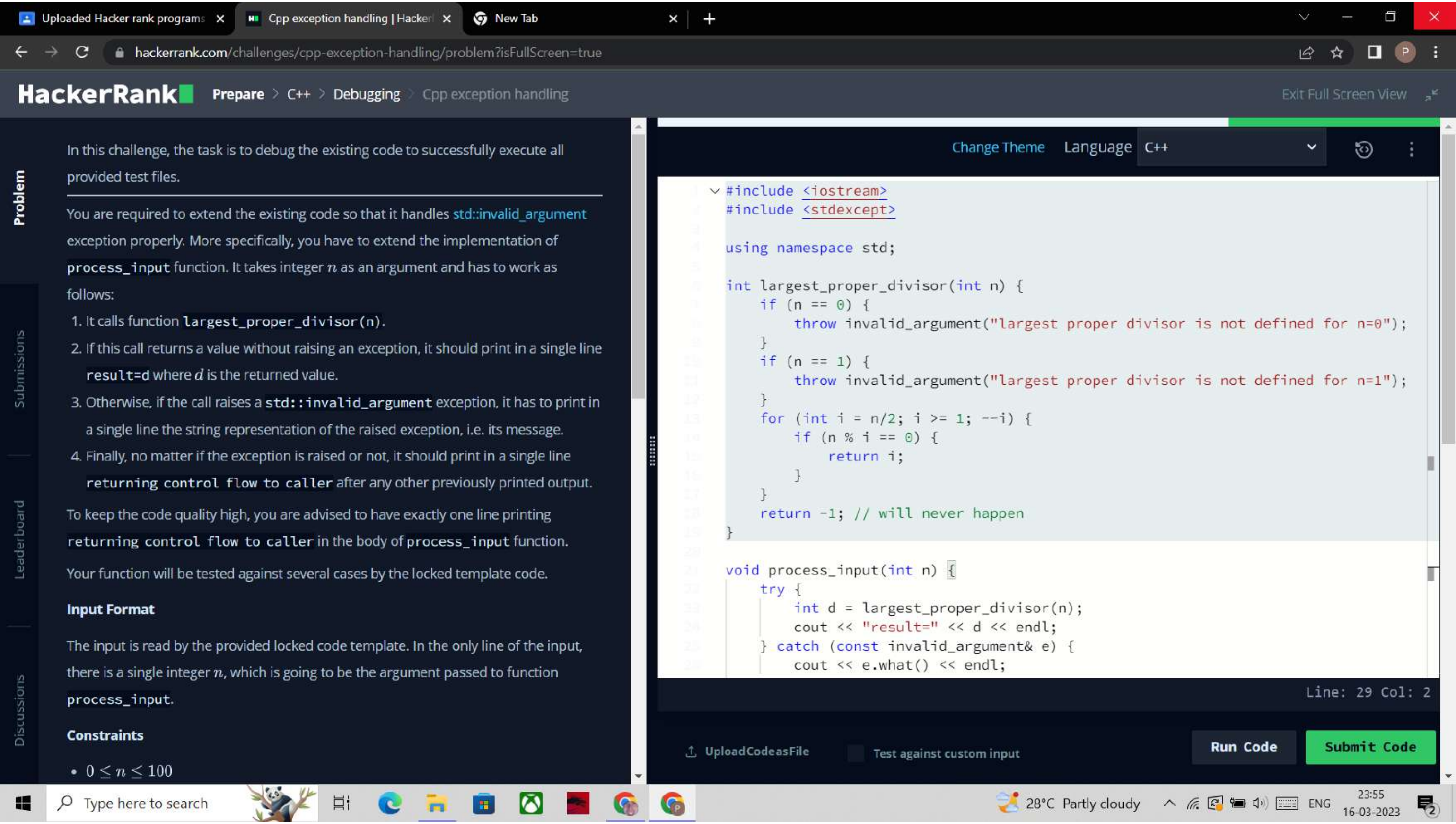
Compiler Message

```
Success
```

Input (stdin)

```
4 3
<tag1 value = "HelloWorld">
<tag2 name = "Name1">
</tag2>
</tag1>
tag1.tag2~name
tag1~name
tag1~value
```

2. Exception Handling



Uploaded Hacker rank programs
Cpp exception handling | Hacker
New Tab
hackerrank.com/challenges/cpp-exception-handling/problem?isFullScreen=true
HackerRank
Prepare > C++ > Debugging > Cpp exception handling
Exit Full Screen View
Problem
Submissions
Leaderboard
Discussions
Change Theme
Language C++
UploadCodeasFile
Test against custom input
Run Code
Submit Code
Type here to search
28°C Partly cloudy
ENG
23:55
16-03-2023

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

You are required to extend the existing code so that it handles std::invalid_argument exception properly. More specifically, you have to extend the implementation of `process_input` function. It takes integer n as an argument and has to work as follows:

1. It calls function `largest_proper_divisor(n)`.
2. If this call returns a value without raising an exception, it should print in a single line `result=d` where d is the returned value.
3. Otherwise, if the call raises a `std::invalid_argument` exception, it has to print in a single line the string representation of the raised exception, i.e. its message.
4. Finally, no matter if the exception is raised or not, it should print in a single line `returning control flow to caller` after any other previously printed output.

To keep the code quality high, you are advised to have exactly one line printing `returning control flow to caller` in the body of `process_input` function.

Your function will be tested against several cases by the locked template code.

Input Format

The input is read by the provided locked code template. In the only line of the input, there is a single integer n, which is going to be the argument passed to function `process_input`.

Constraints

- $0 \leq n \leq 100$

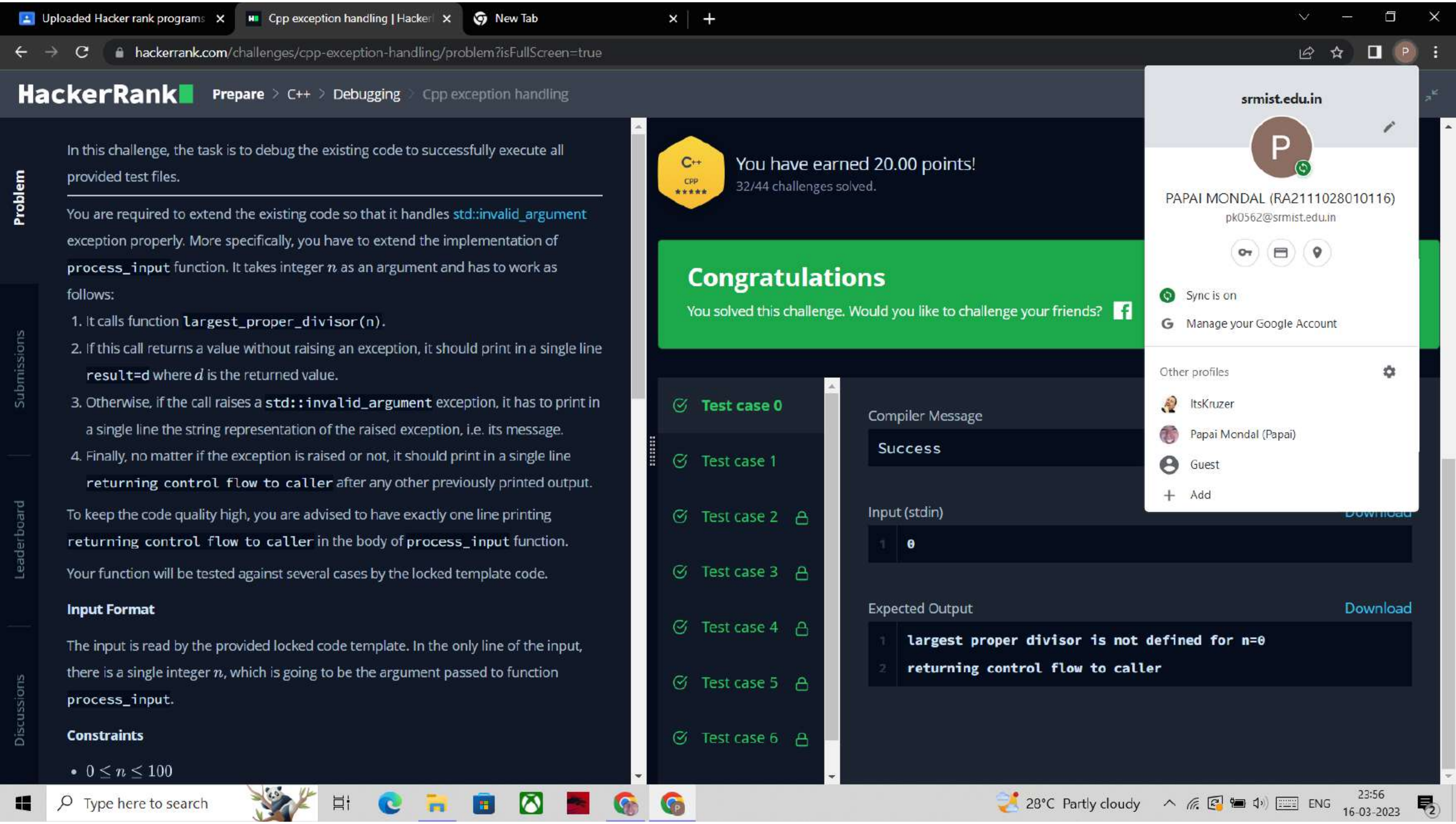
```
#include <iostream>
#include <stdexcept>

using namespace std;

int largest_proper_divisor(int n) {
    if (n == 0) {
        throw invalid_argument("largest proper divisor is not defined for n=0");
    }
    if (n == 1) {
        throw invalid_argument("largest proper divisor is not defined for n=1");
    }
    for (int i = n/2; i >= 1; --i) {
        if (n % i == 0) {
            return i;
        }
    }
    return -1; // will never happen
}

void process_input(int n) {
    try {
        int d = largest_proper_divisor(n);
        cout << "result=" << d << endl;
    } catch (const invalid_argument& e) {
        cout << e.what() << endl;
```

Line: 29 Col: 2


Uploaded Hacker rank programs
Cpp exception handling | HackerR
New Tab
hackerrank.com/challenges/cpp-exception-handling/problem?isFullScreen=true
HackerRank
Prepare > C++ > Debugging > Cpp exception handling
Problem
Submissions
Leaderboard
Discussions

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

You are required to extend the existing code so that it handles std::invalid_argument exception properly. More specifically, you have to extend the implementation of `process_input` function. It takes integer n as an argument and has to work as follows:

1. It calls function `largest_proper_divisor(n)`.
2. If this call returns a value without raising an exception, it should print in a single line `result=d` where d is the returned value.
3. Otherwise, if the call raises a `std::invalid_argument` exception, it has to print in a single line the string representation of the raised exception, i.e. its message.
4. Finally, no matter if the exception is raised or not, it should print in a single line `returning control flow to caller` after any other previously printed output.

To keep the code quality high, you are advised to have exactly one line printing `returning control flow to caller` in the body of `process_input` function.

Your function will be tested against several cases by the locked template code.

Input Format

The input is read by the provided locked code template. In the only line of the input, there is a single integer n, which is going to be the argument passed to function `process_input`.

Constraints

- $0 \leq n \leq 100$

You have earned 20.00 points!
32/44 challenges solved.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Compiler Message

```
Success
```

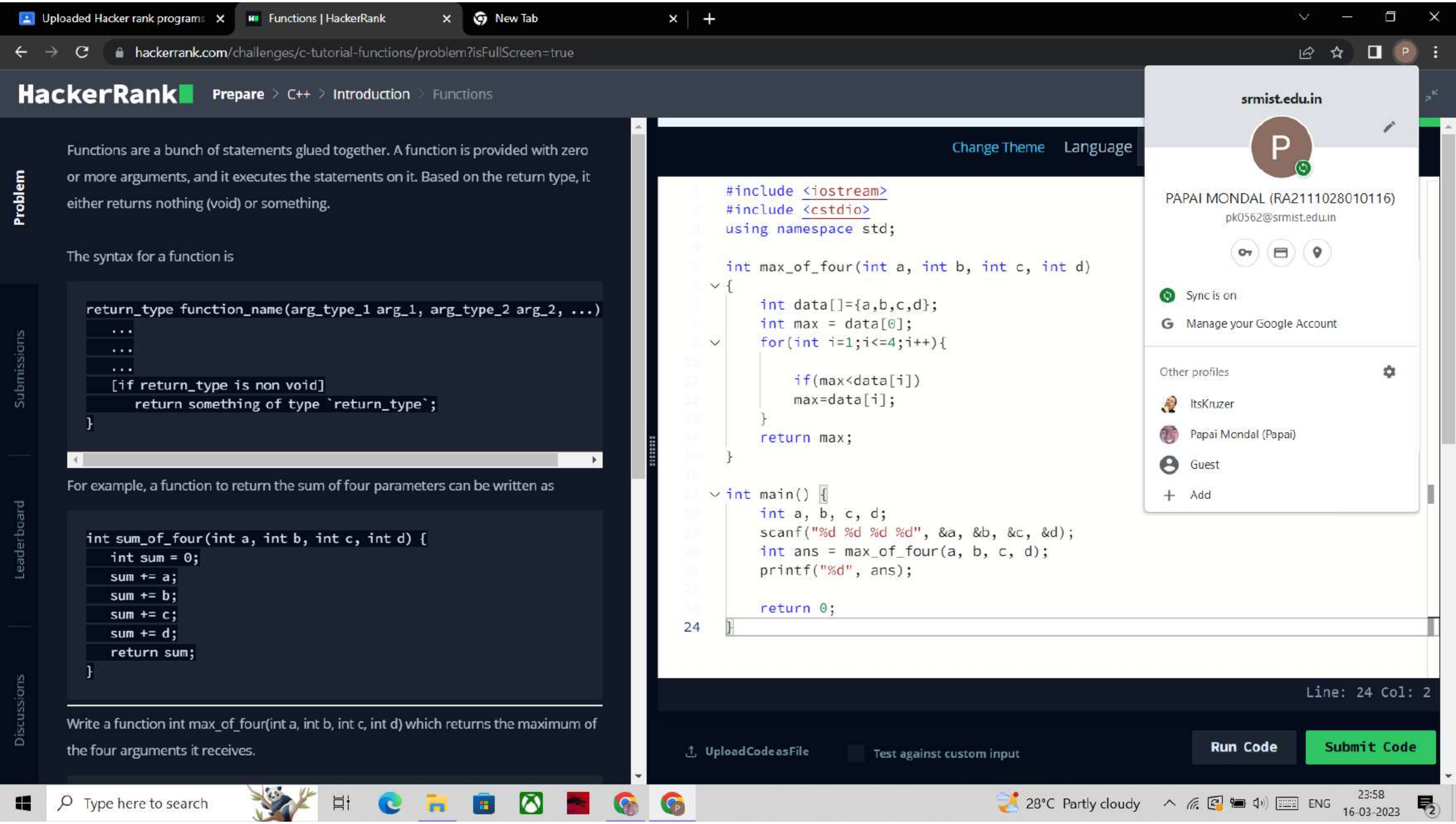
Input (stdin)

```
0
```

Expected Output

```
largest proper divisor is not defined for n=0
returning control flow to caller
```

3. Functions



Functions are a bunch of statements glued together. A function is provided with zero or more arguments, and it executes the statements on it. Based on the return type, it either returns nothing (void) or something.

The syntax for a function is

```
return_type function_name(arg_type_1 arg_1, arg_type_2 arg_2, ...)
    ...
    ...
    ...
    [if return_type is non void]
        return something of type `return_type`;
}
```

For example, a function to return the sum of four parameters can be written as

```
int sum_of_four(int a, int b, int c, int d) {
    int sum = 0;
    sum += a;
    sum += b;
    sum += c;
    sum += d;
    return sum;
}
```

Write a function int max_of_four(int a, int b, int c, int d) which returns the maximum of the four arguments it receives.

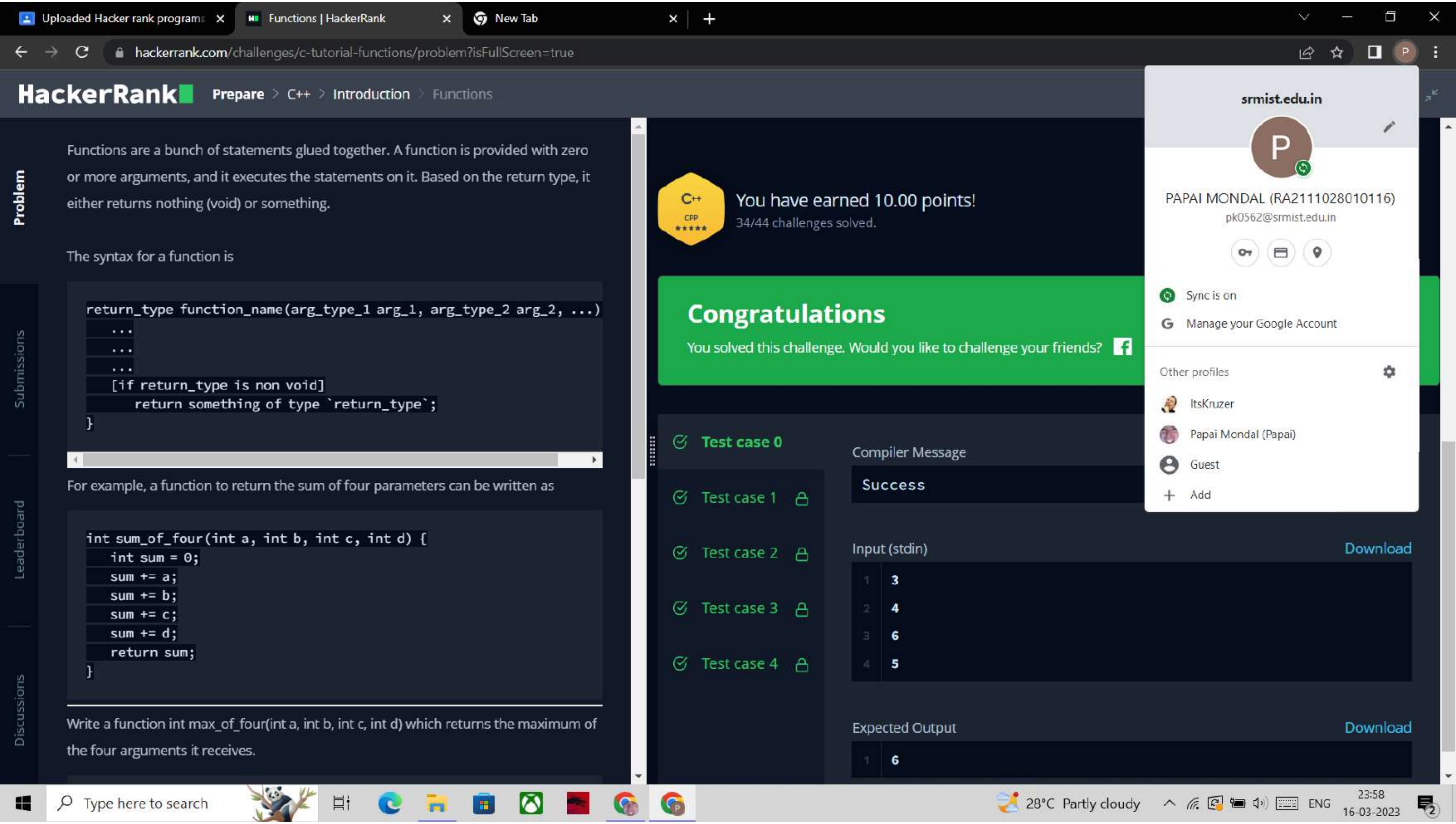
```
#include <iostream>
#include <cstdio>
using namespace std;

int max_of_four(int a, int b, int c, int d)
{
    int data[]={a,b,c,d};
    int max = data[0];
    for(int i=1;i<=4;i++){

        if(max<data[i])
        max=data[i];
    }
    return max;
}

int main() {
    int a, b, c, d;
    scanf("%d %d %d %d", &a, &b, &c, &d);
    int ans = max_of_four(a, b, c, d);
    printf("%d", ans);

    return 0;
}
```


Uploaded Hacker rank programs
Functions | HackerRank
New Tab
hackerrank.com/challenges/c-tutorial-functions/problem?isFullScreen=true
HackerRank
Prepare > C++ > Introduction > Functions
Problem
Submissions
Leaderboard
Discussions

Functions are a bunch of statements glued together. A function is provided with zero or more arguments, and it executes the statements on it. Based on the return type, it either returns nothing (void) or something.

The syntax for a function is

```
return_type function_name(arg_type_1 arg_1, arg_type_2 arg_2, ...)
    ...
    ...
    ...
    [if return_type is non void]
        return something of type `return_type`;
}
```

For example, a function to return the sum of four parameters can be written as

```
int sum_of_four(int a, int b, int c, int d) {
    int sum = 0;
    sum += a;
    sum += b;
    sum += c;
    sum += d;
    return sum;
}
```

Write a function int max_of_four(int a, int b, int c, int d) which returns the maximum of the four arguments it receives.

You have earned 10.00 points!
34/44 challenges solved.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Compiler Message

```
Success
```

Input (stdin)

```
3
4
6
5
```

Expected Output

```
6
```