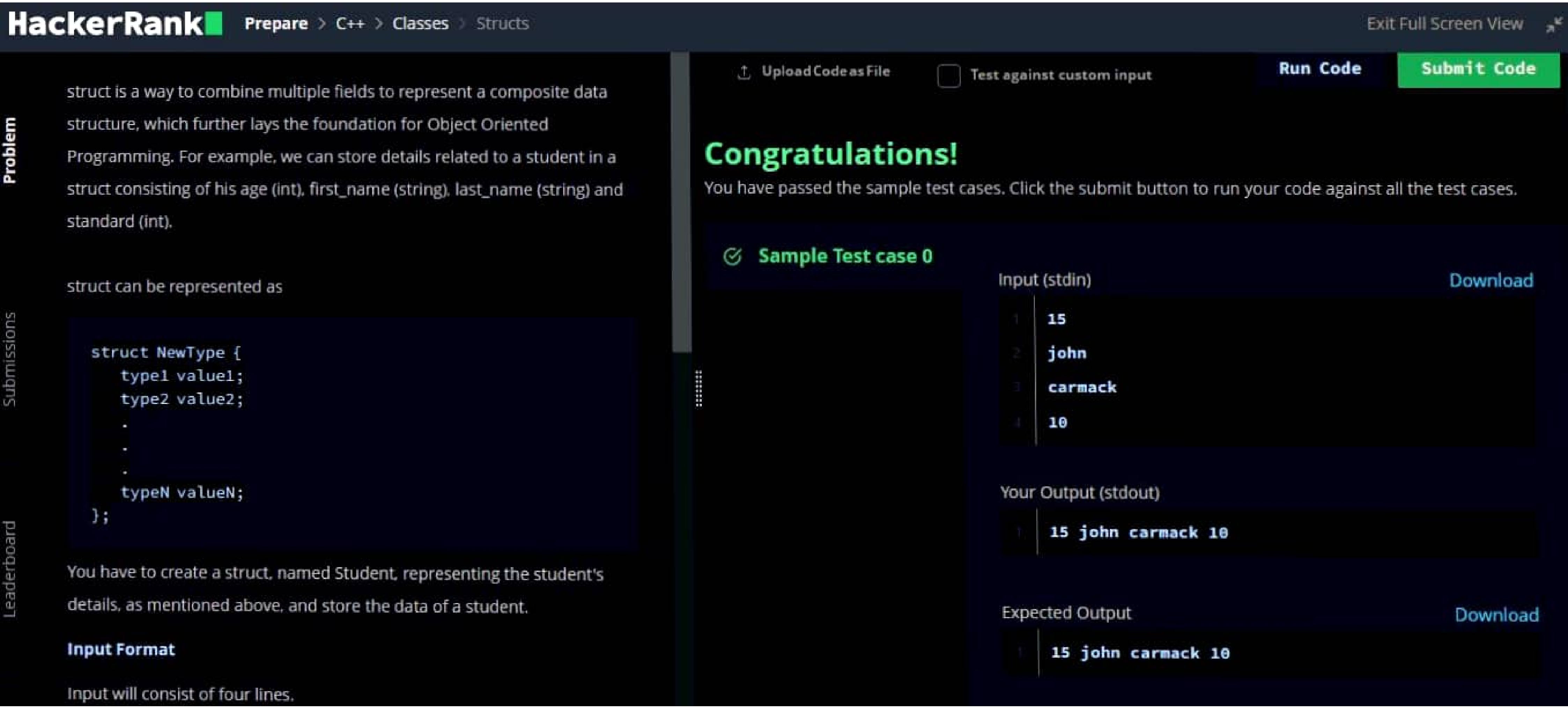


struct is a way to combine multiple fields to represent a composite data structure, which further lays the foundation for Object Oriented Programming. For example, we can store details related to a student in a struct consisting of his age (int), first_name (string), last_name (string) and standard (int).

struct can be represented as

```
struct NewType {
    type1 value1;
    type2 value2;
    .
    .
    .
    typeN valueN;
};
```

You have to create a struct, named Student, representing the student's details, as mentioned above, and store the data of a student.

Input Format

Input will consist of four lines.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
15
john
carmack
10
```

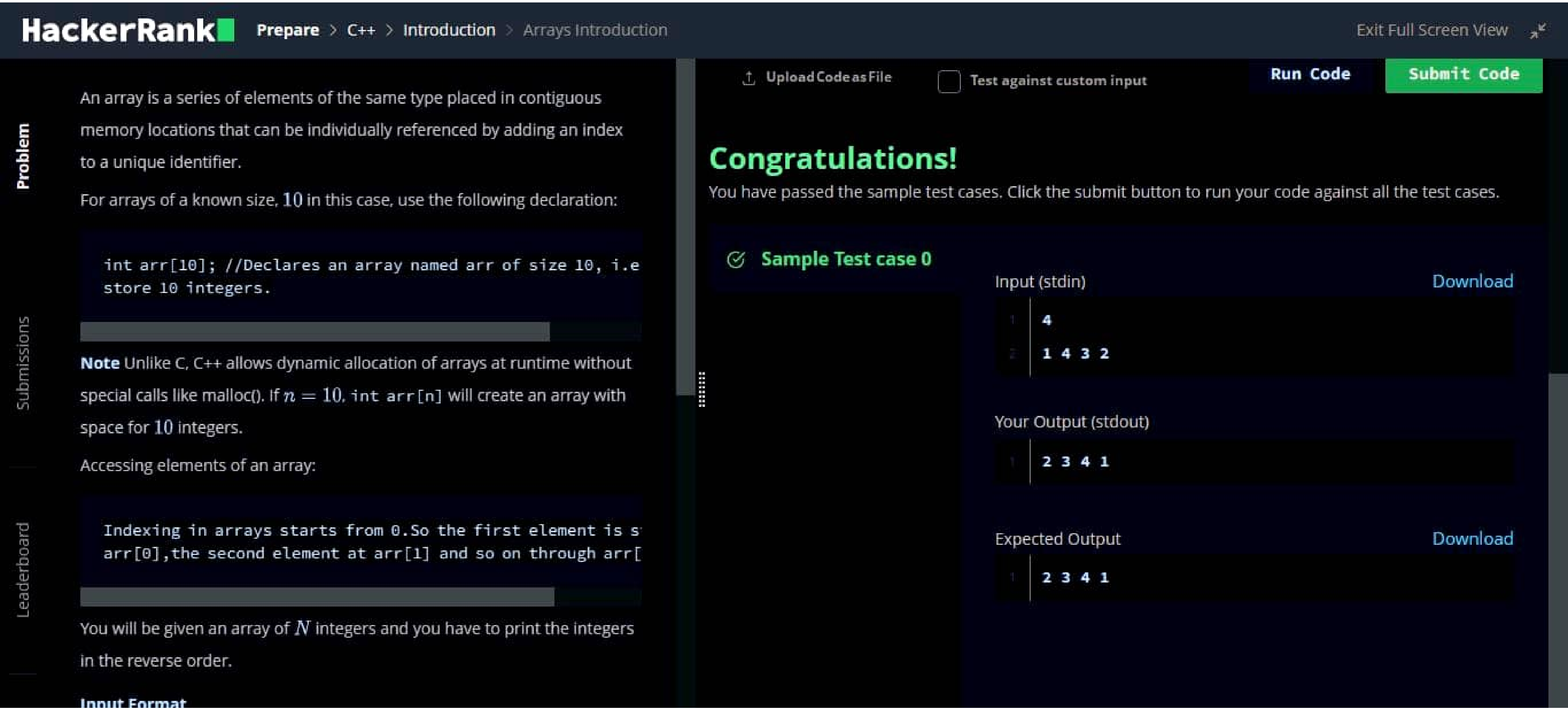
Your Output (stdout)

```
15 john carmack 10
```

Expected Output Download

```
15 john carmack 10
```

Scanned with CamScanner


HackerRank Prepare > C++ > Introduction > Arrays Introduction Exit Full Screen View

Problem Submissions Leaderboard

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.

For arrays of a known size, 10 in this case, use the following declaration:

```
int arr[10]; //Declares an array named arr of size 10, i.e
store 10 integers.
```

Note Unlike C, C++ allows dynamic allocation of arrays at runtime without special calls like malloc(). If $n = 10$, `int arr[n]` will create an array with space for 10 integers.

Accessing elements of an array:

```
Indexing in arrays starts from 0.So the first element is s
arr[0],the second element at arr[1] and so on through arr[
```

You will be given an array of N integers and you have to print the integers in the reverse order.

Input Format

Upload Code as File   Test against custom input   Run Code   Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

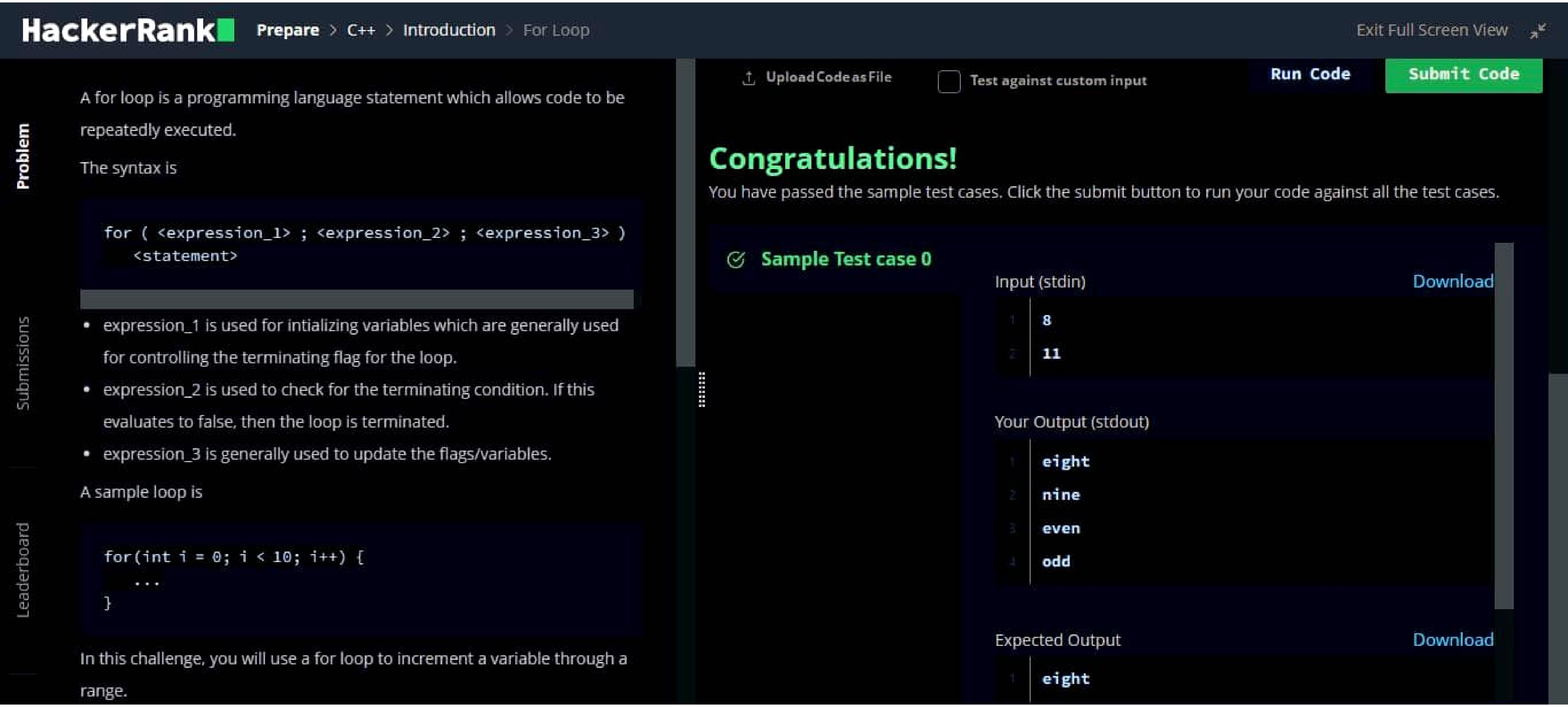
```
4
1 4 3 2
```

Your Output (stdout)

```
2 3 4 1
```

Expected Output Download

```
2 3 4 1
```



A for loop is a programming language statement which allows code to be repeatedly executed.

The syntax is

```
for ( <expression_1> ; <expression_2> ; <expression_3> )
    <statement>
```

- expression_1 is used for intializing variables which are generally used for controlling the terminating flag for the loop.
- expression_2 is used to check for the terminating condition. If this evaluates to false, then the loop is terminated.
- expression_3 is generally used to update the flags/variables.

A sample loop is

```
for(int i = 0; i < 10; i++) {
    ...
}
```

In this challenge, you will use a for loop to increment a variable through a range.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

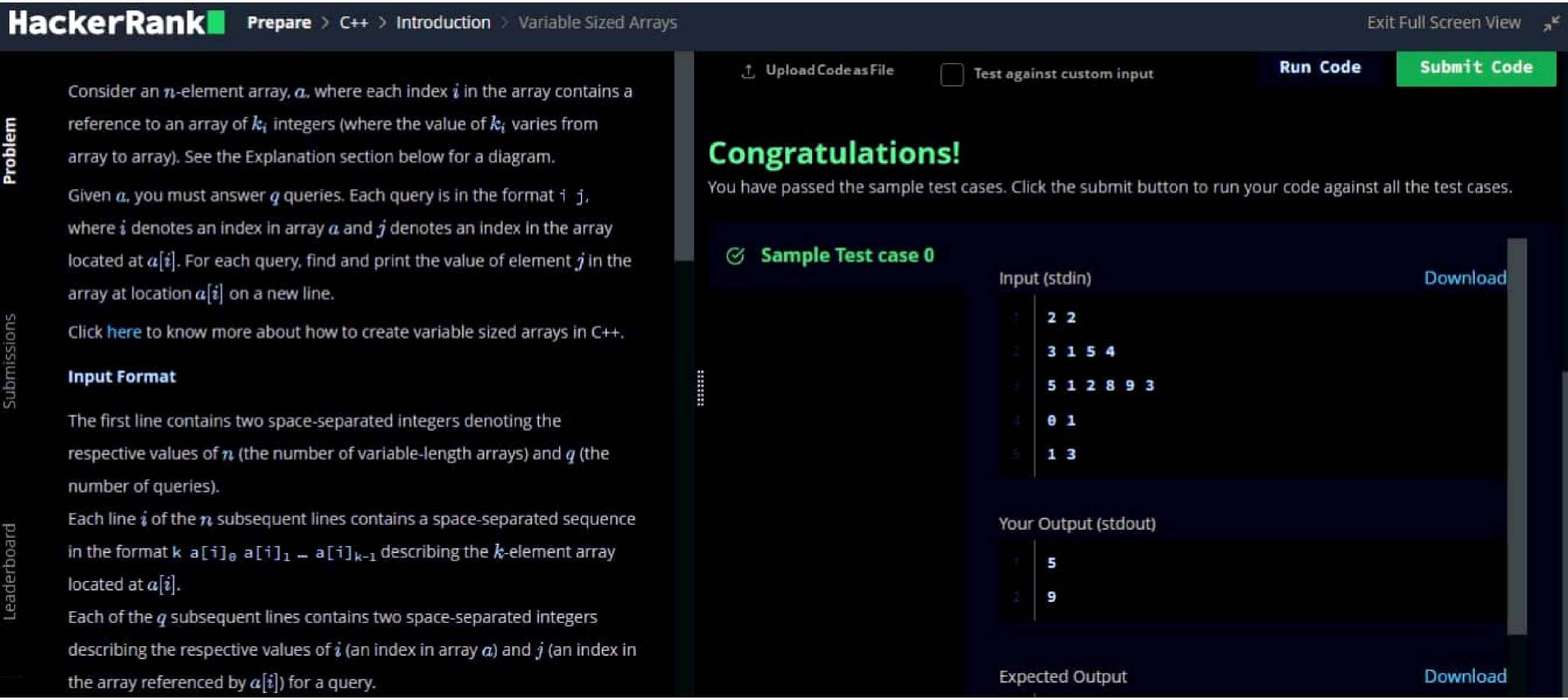
```
8
11
```

Your Output (stdout)

```
eight
nine
even
odd
```

Expected Output Download

```
eight
```



Consider an n-element array, a, where each index i in the array contains a reference to an array of k_i integers (where the value of k_i varies from array to array). See the Explanation section below for a diagram.

Given a, you must answer q queries. Each query is in the format `i j`, where i denotes an index in array a and j denotes an index in the array located at $a[i]$. For each query, find and print the value of element j in the array at location $a[i]$ on a new line.

Click here to know more about how to create variable sized arrays in C++.

Input Format

The first line contains two space-separated integers denoting the respective values of n (the number of variable-length arrays) and q (the number of queries).

Each line i of the n subsequent lines contains a space-separated sequence in the format `k a[i]`$_0$ `a[i]`$_1$ … `a[i]`$_{k-1}$ describing the k-element array located at $a[i]$.

Each of the q subsequent lines contains two space-separated integers describing the respective values of i (an index in array a) and j (an index in the array referenced by $a[i]$) for a query.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

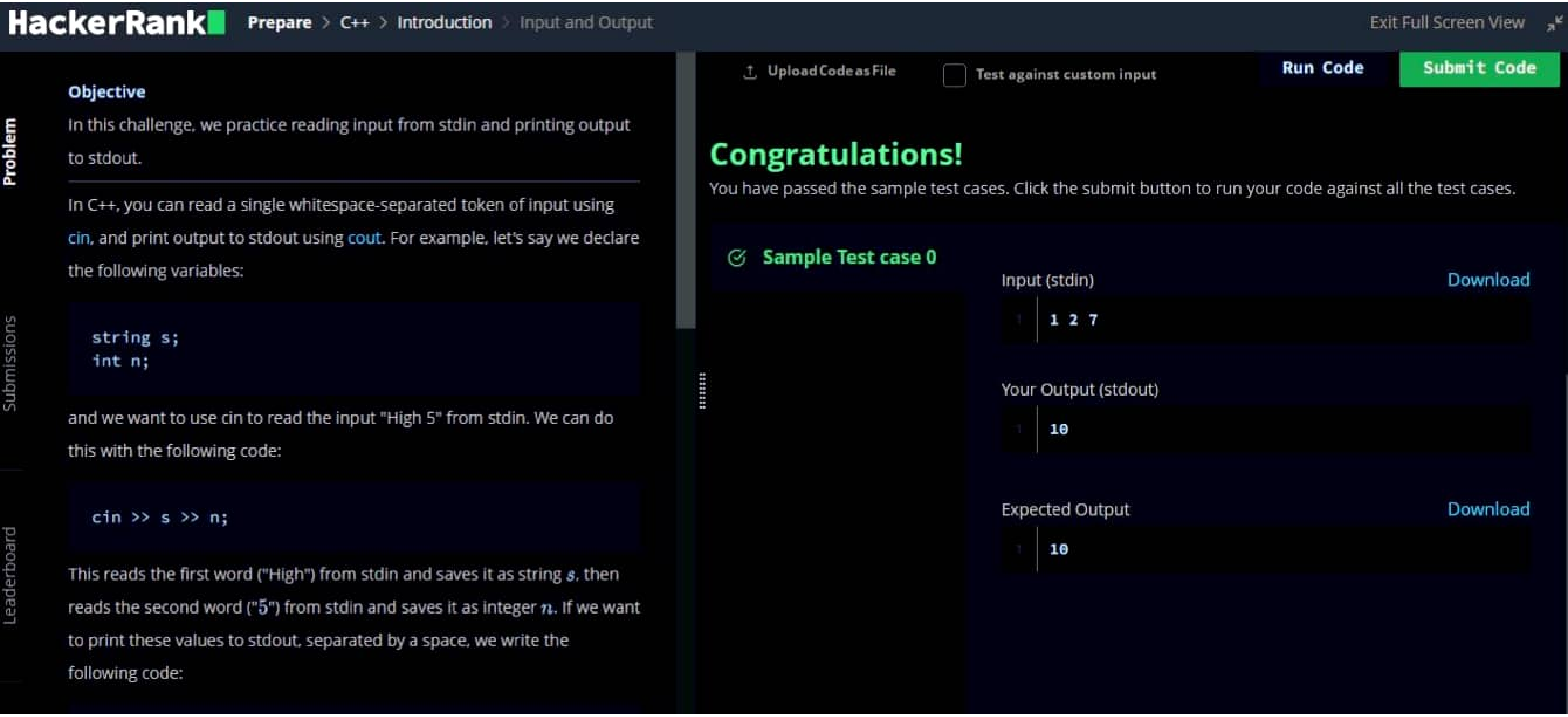
```
2 2
3 1 5 4
5 1 2 8 9 3
0 1
1 3
```

Your Output (stdout)

```
5
9
```

Expected Output Download

Scanned with CamScanner



Objective

In this challenge, we practice reading input from stdin and printing output to stdout.

In C++, you can read a single whitespace-separated token of input using cin, and print output to stdout using cout. For example, let's say we declare the following variables:

```
string s;
int n;
```

and we want to use cin to read the input "High 5" from stdin. We can do this with the following code:

```
cin >> s >> n;
```

This reads the first word ("High") from stdin and saves it as string s, then reads the second word ("5") from stdin and saves it as integer n. If we want to print these values to stdout, separated by a space, we write the following code:

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

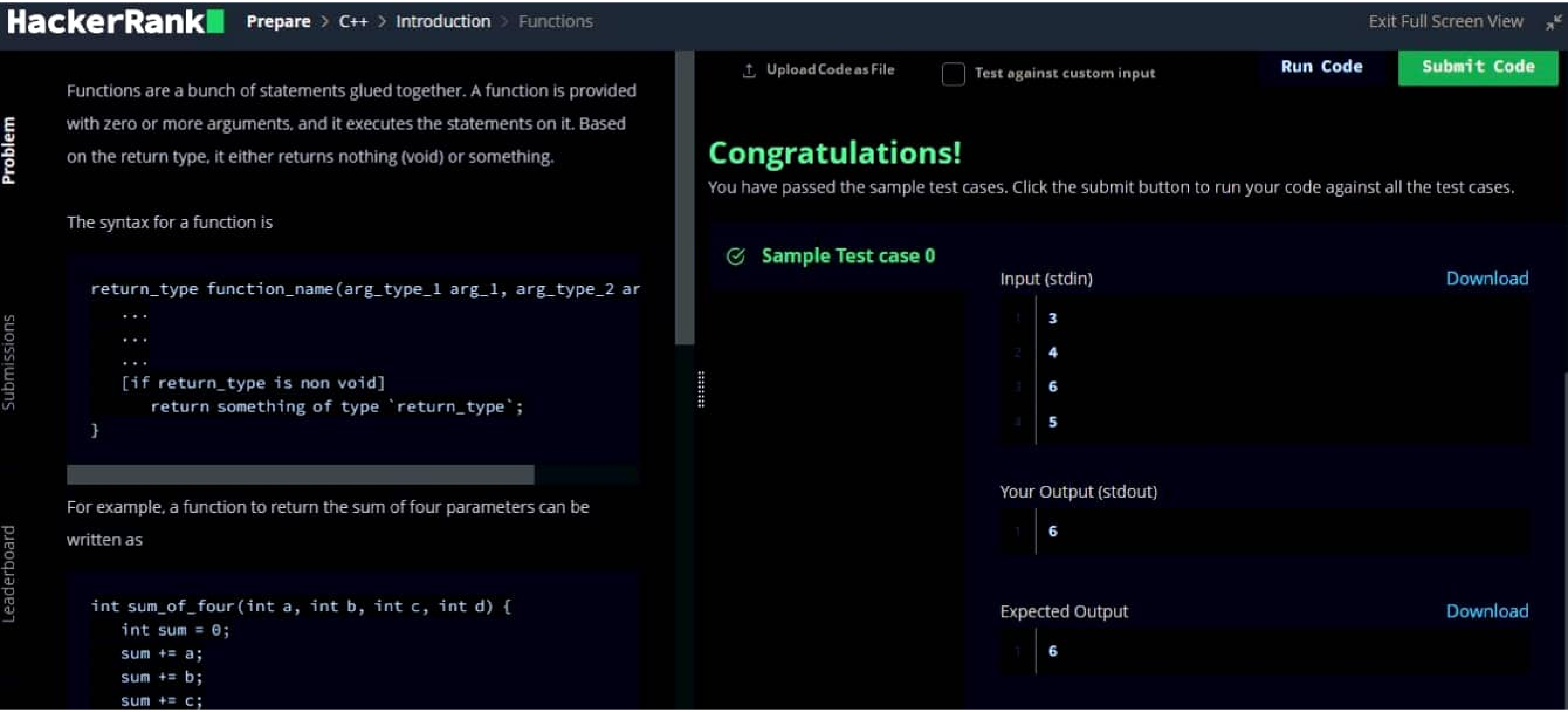
```
1 2 7
```

Your Output (stdout)

```
10
```

Expected Output Download

```
10
```



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 ar
    ...
    ...
    ...
    [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;
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
3
4
6
5
```

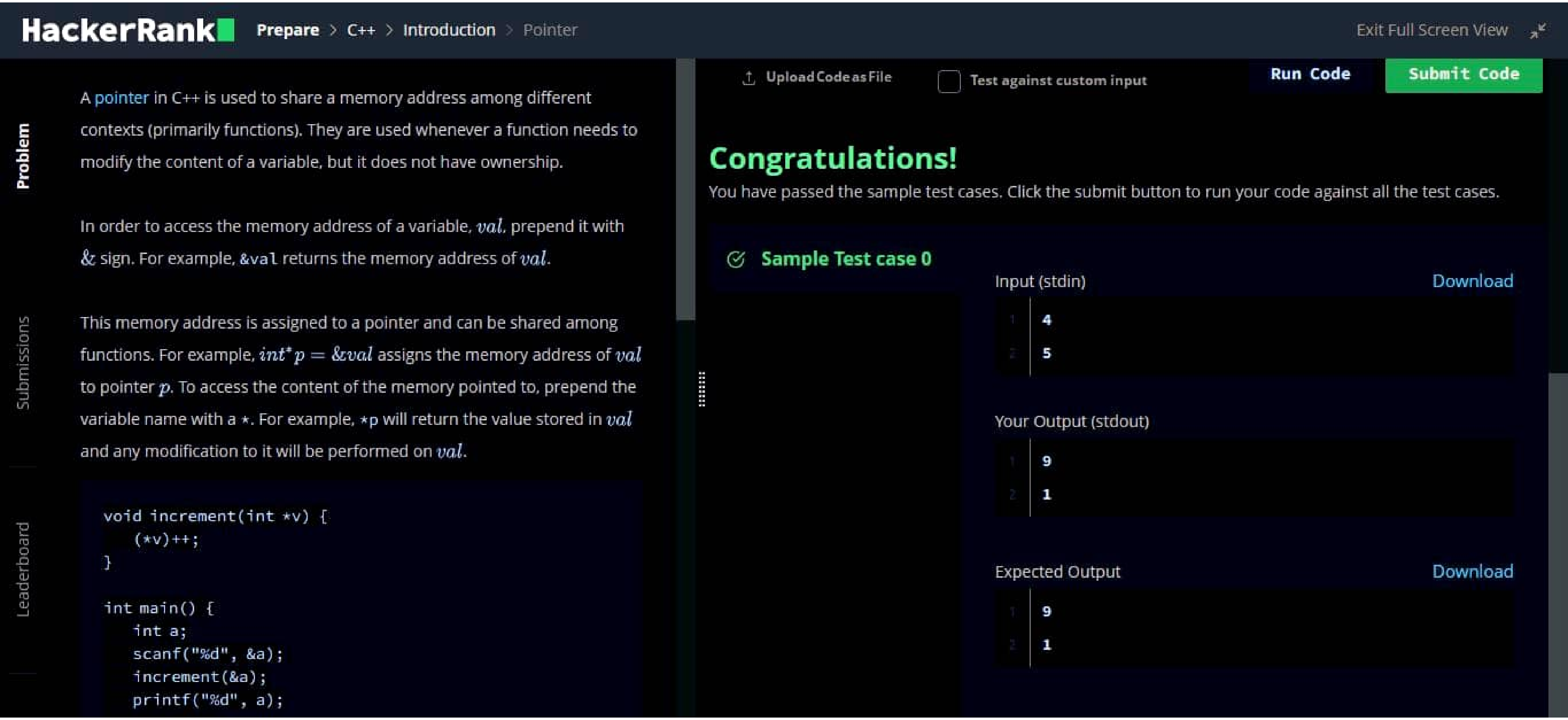
Your Output (stdout)

```
6
```

Expected Output Download

```
6
```

Scanned with CamScanner



A pointer in C++ is used to share a memory address among different contexts (primarily functions). They are used whenever a function needs to modify the content of a variable, but it does not have ownership.

In order to access the memory address of a variable, val, prepend it with $\&$ sign. For example, `&val` returns the memory address of val.

This memory address is assigned to a pointer and can be shared among functions. For example, $int^{*}p = \&val$ assigns the memory address of val to pointer p. To access the content of the memory pointed to, prepend the variable name with a `*`. For example, `*p` will return the value stored in val and any modification to it will be performed on val.

```
void increment(int *v) {
    (*v)++;
}

int main() {
    int a;
    scanf("%d", &a);
    increment(&a);
    printf("%d", a);
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

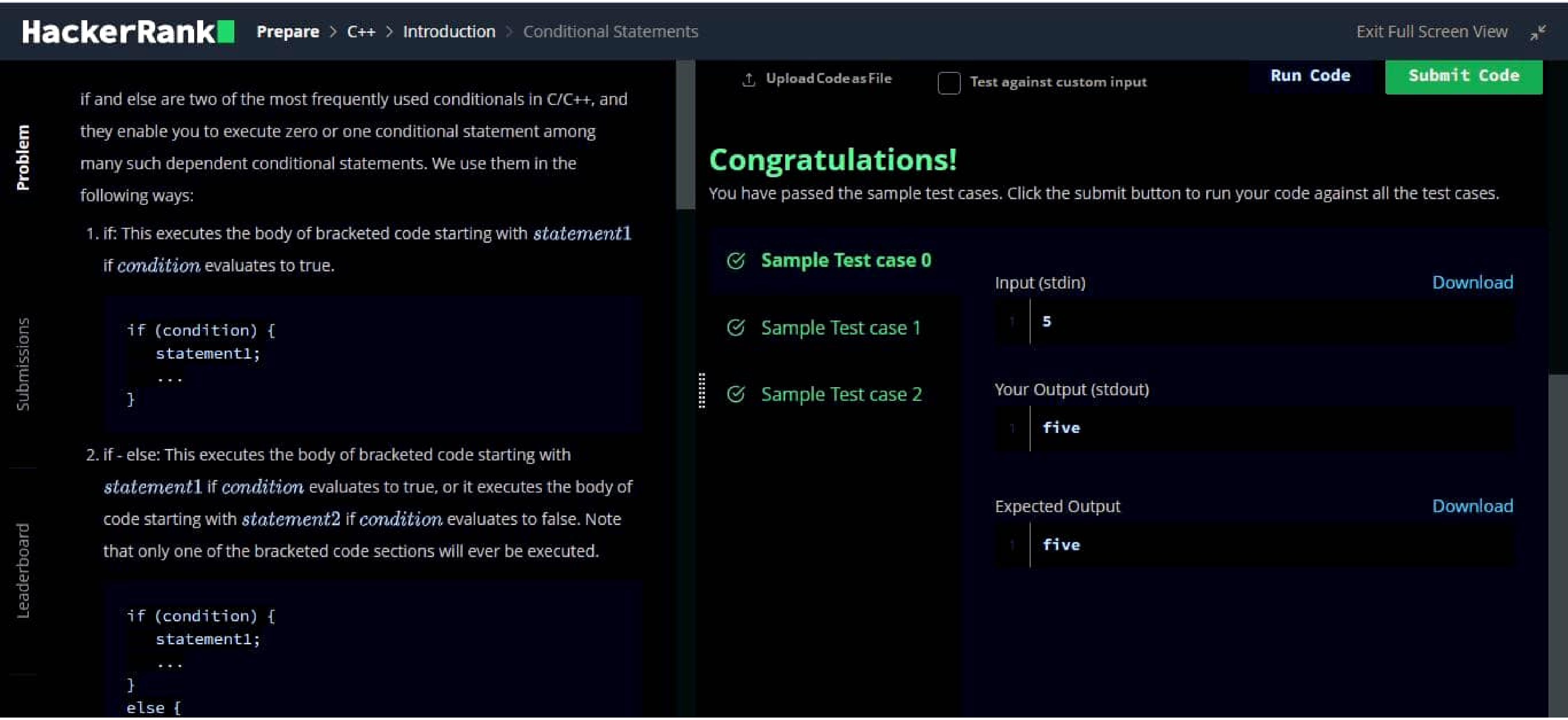
```
4
5
```

Your Output (stdout)

```
9
1
```

Expected Output Download

```
9
1
```


HackerRank Prepare > C++ > Introduction > Conditional Statements

Exit Full Screen View

Problem

Submissions

Leaderboard

if and else are two of the most frequently used conditionals in C/C++, and they enable you to execute zero or one conditional statement among many such dependent conditional statements. We use them in the following ways:

1. if: This executes the body of bracketed code starting with $statement1$ if $condition$ evaluates to true.

```
if (condition) {
    statement1;
    ...
}
```

2. if - else: This executes the body of bracketed code starting with $statement1$ if $condition$ evaluates to true, or it executes the body of code starting with $statement2$ if $condition$ evaluates to false. Note that only one of the bracketed code sections will ever be executed.

```
if (condition) {
    statement1;
    ...
}
else {
```

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

✓ Sample Test case 1

✓ Sample Test case 2

Input (stdin) Download

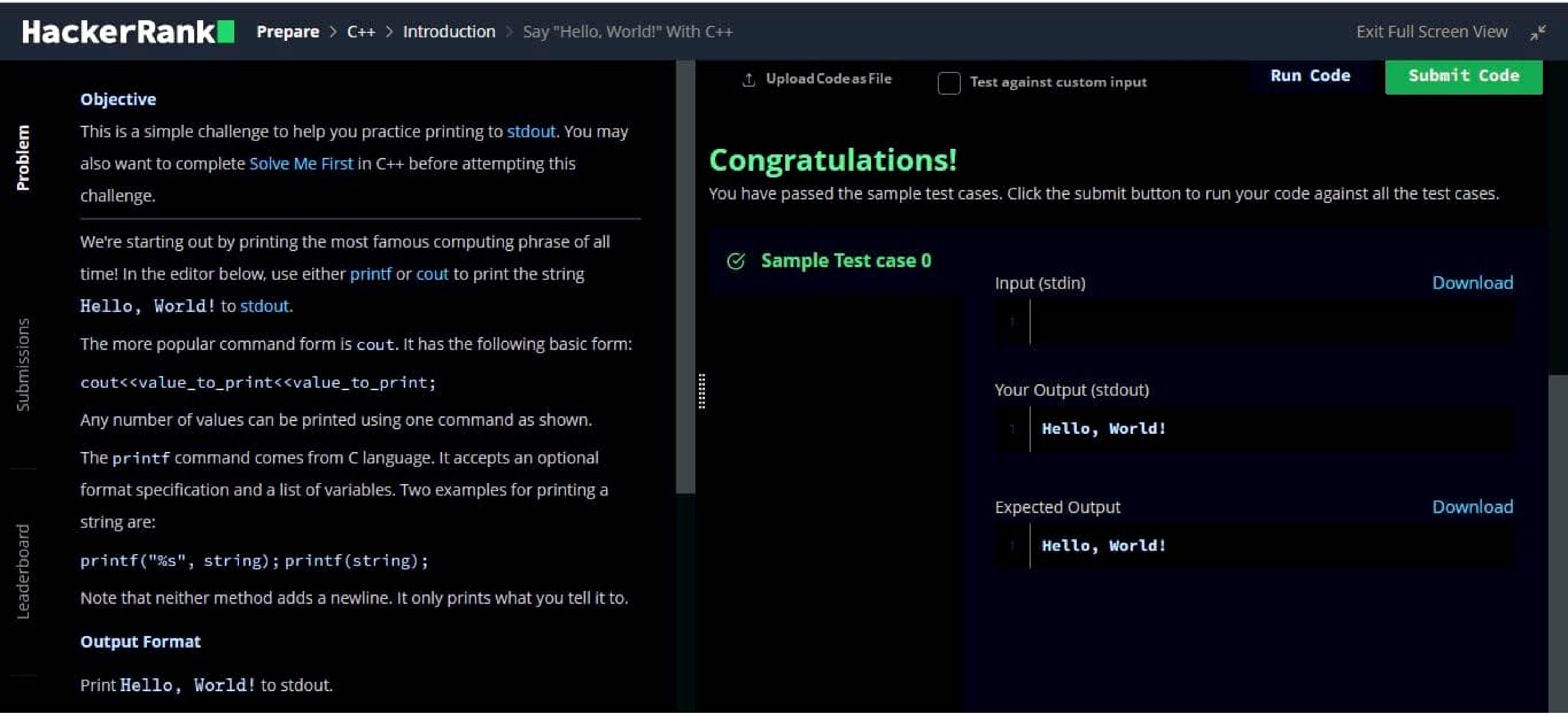
```
5
```

Your Output (stdout)

```
five
```

Expected Output Download

```
five
```



Objective

This is a simple challenge to help you practice printing to stdout. You may also want to complete Solve Me First in C++ before attempting this challenge.

We're starting out by printing the most famous computing phrase of all time! In the editor below, use either printf or cout to print the string `Hello, World!` to stdout.

The more popular command form is `cout`. It has the following basic form:

```
cout<<value_to_print<<value_to_print;
```

Any number of values can be printed using one command as shown.

The `printf` command comes from C language. It accepts an optional format specification and a list of variables. Two examples for printing a string are:

```
printf("%s", string); printf(string);
```

Note that neither method adds a newline. It only prints what you tell it to.

Output Format

Print `Hello, World!` to stdout.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

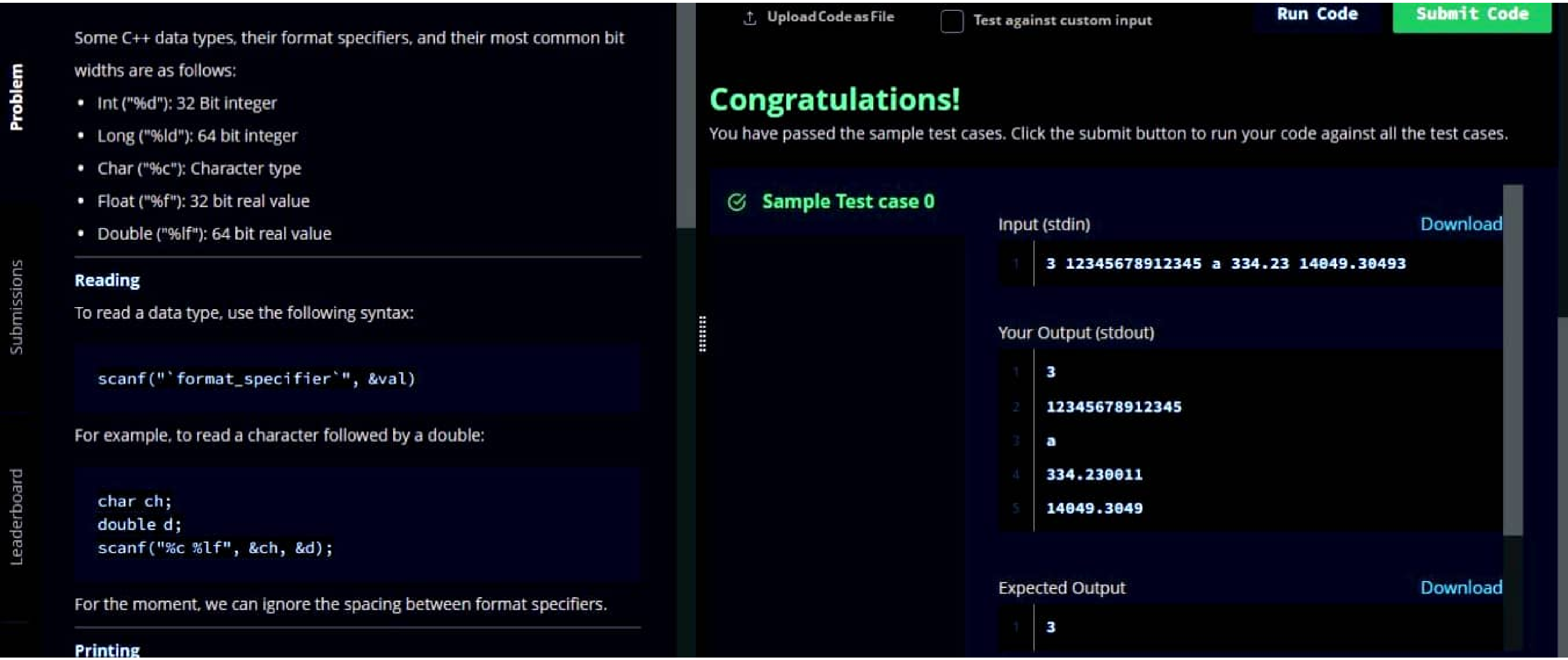
Input (stdin) Download

Your Output (stdout)

```
Hello, World!
```

Expected Output Download

```
Hello, World!
```



Some C++ data types, their format specifiers, and their most common bit widths are as follows:

- Int ("%d"): 32 Bit integer
- Long ("%ld"): 64 bit integer
- Char ("%c"): Character type
- Float ("%f"): 32 bit real value
- Double ("%lf"): 64 bit real value

Reading

To read a data type, use the following syntax:

```
scanf("`format_specifier`", &val)
```

For example, to read a character followed by a double:

```
char ch;
double d;
scanf("%c %lf", &ch, &d);
```

For the moment, we can ignore the spacing between format specifiers.

Printing

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
3 12345678912345 a 334.23 14049.30493
```

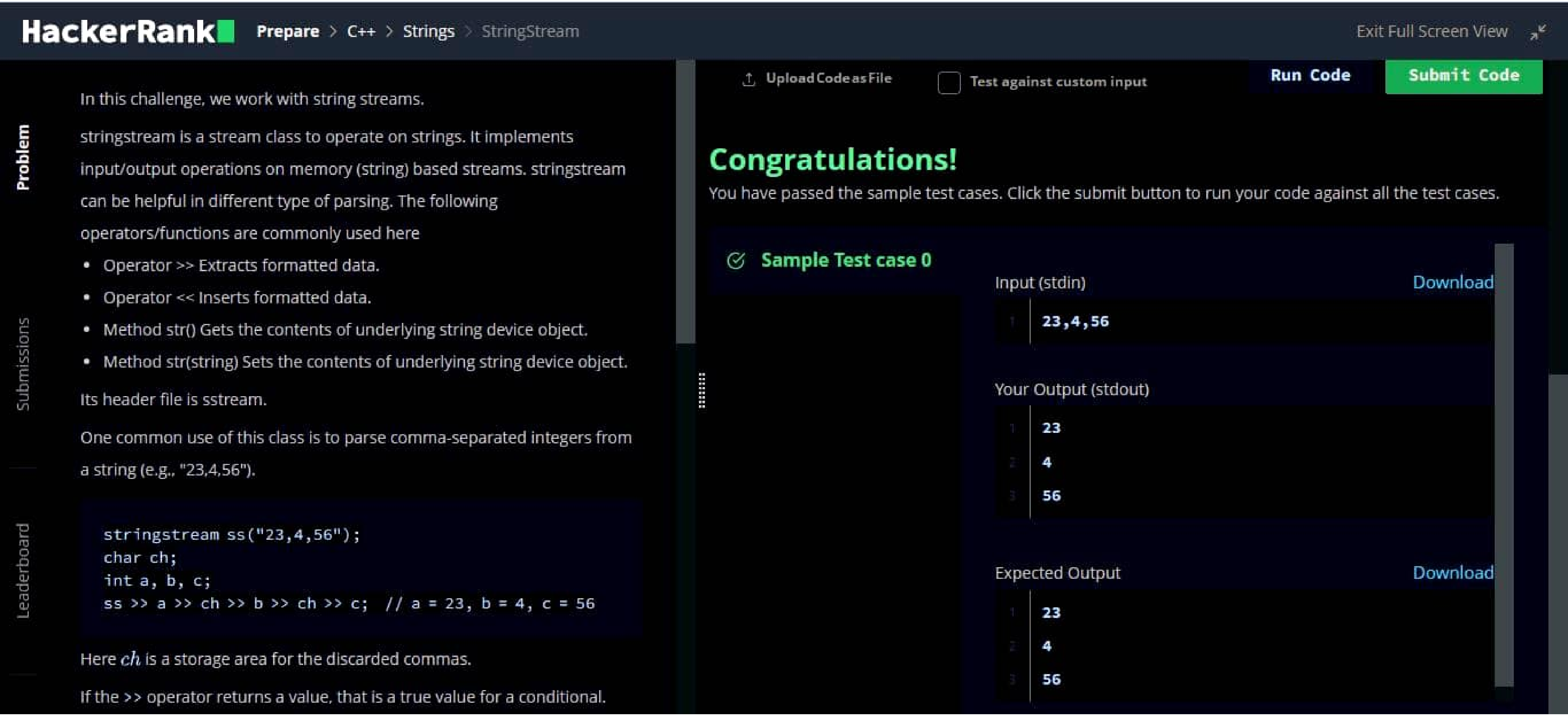
Your Output (stdout)

```
3
12345678912345
a
334.230011
14049.3049
```

Expected Output Download

```
3
```

Scanned with CamScanner



In this challenge, we work with string streams.

stringstream is a stream class to operate on strings. It implements input/output operations on memory (string) based streams. stringstream can be helpful in different type of parsing. The following operators/functions are commonly used here

- Operator >> Extracts formatted data.
- Operator << Inserts formatted data.
- Method str() Gets the contents of underlying string device object.
- Method str(string) Sets the contents of underlying string device object.

Its header file is sstream.

One common use of this class is to parse comma-separated integers from a string (e.g., "23,4,56").

```
stringstream ss("23,4,56");
char ch;
int a, b, c;
ss >> a >> ch >> b >> ch >> c;  // a = 23, b = 4, c = 56
```

Here ch is a storage area for the discarded commas.

If the >> operator returns a value, that is a true value for a conditional.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

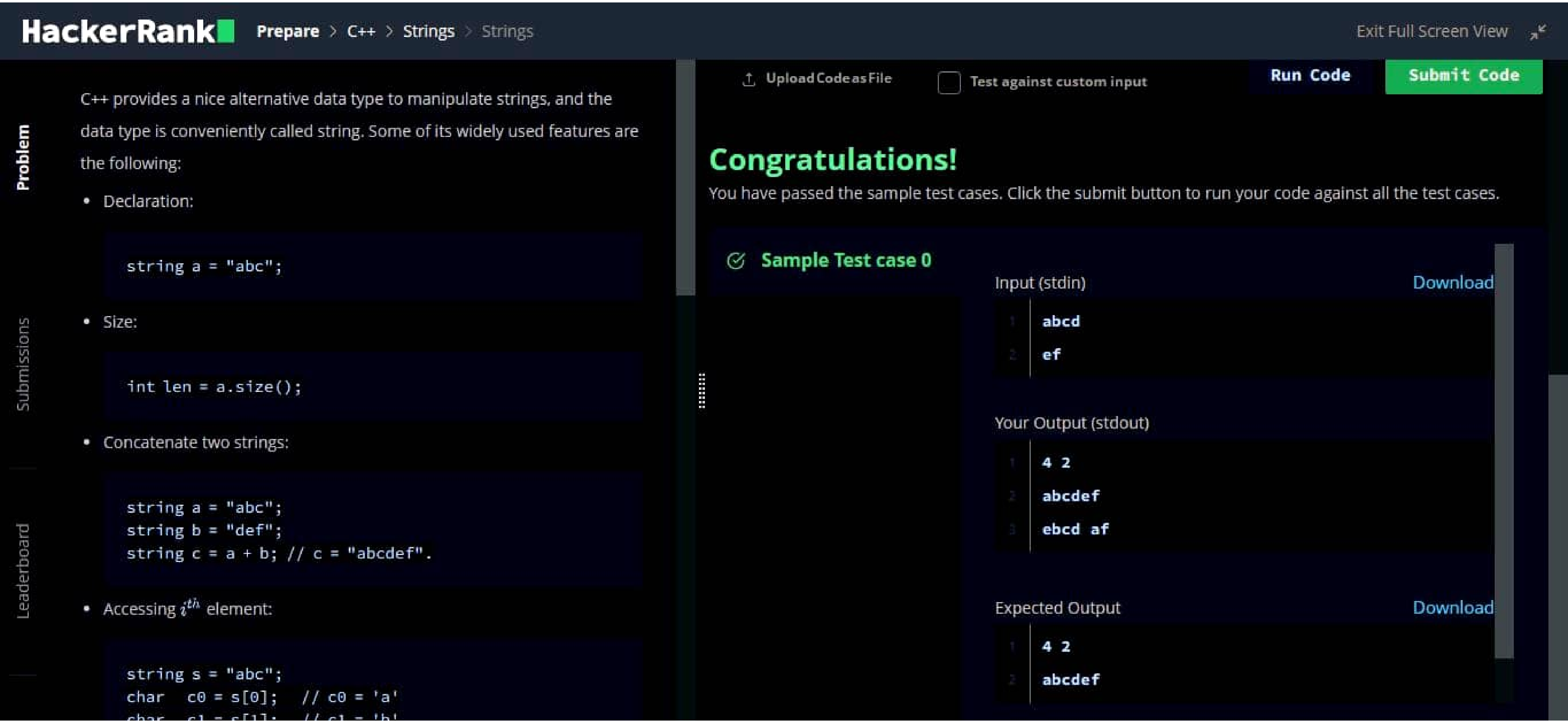
```
23,4,56
```

Your Output (stdout)

```
23
4
56
```

Expected Output Download

```
23
4
56
```



C++ provides a nice alternative data type to manipulate strings, and the data type is conveniently called string. Some of its widely used features are the following:

- Declaration:

```
string a = "abc";
```

- Size:

```
int len = a.size();
```

- Concatenate two strings:

```
string a = "abc";
string b = "def";
string c = a + b; // c = "abcdef".
```

- Accessing i^{th} element:

```
string s = "abc";
char  c0 = s[0];   // c0 = 'a'
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)

```
abcd
ef
```

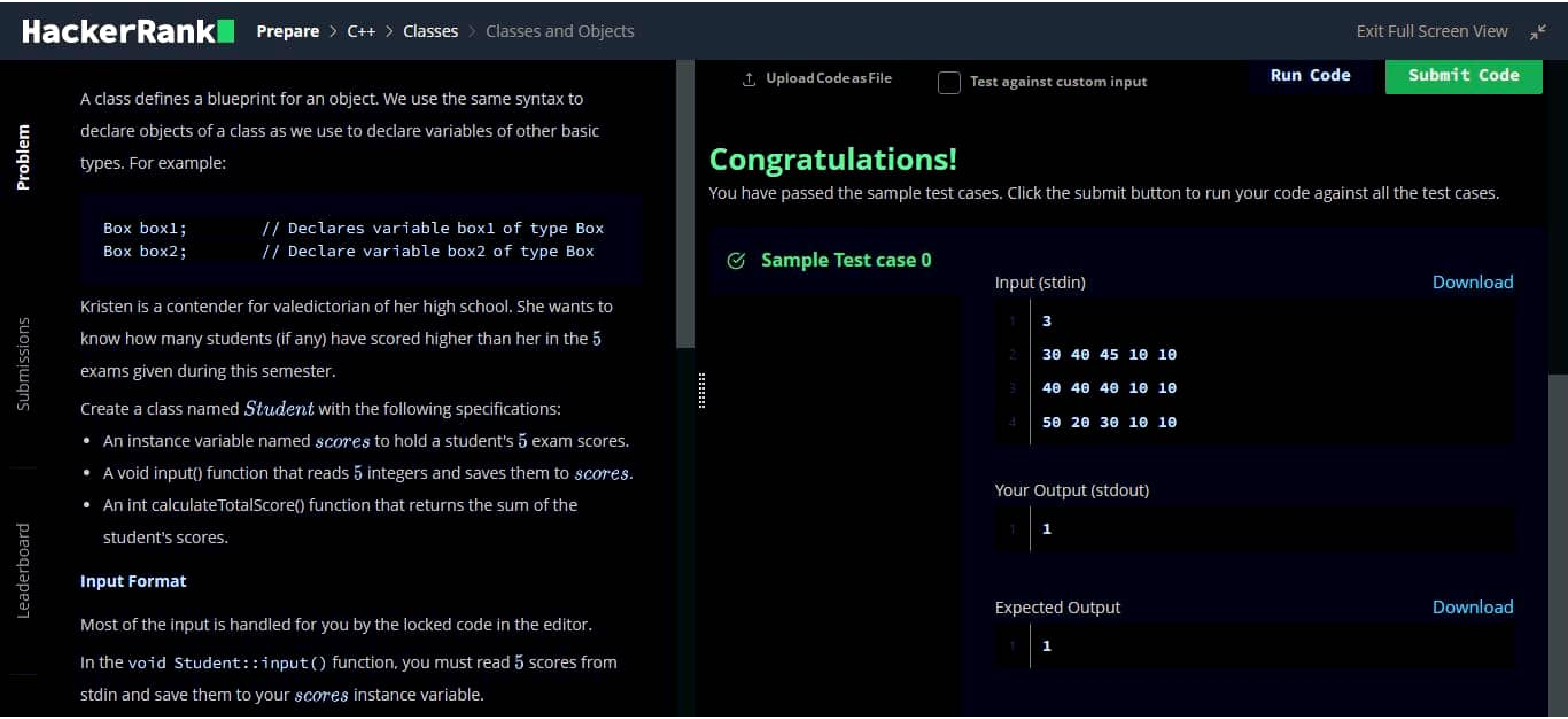
Your Output (stdout)

```
4 2
abcdef
ebcd af
```

Expected Output

```
4 2
abcdef
```

Scanned with CamScanner



A class defines a blueprint for an object. We use the same syntax to declare objects of a class as we use to declare variables of other basic types. For example:

```
Box box1;         // Declares variable box1 of type Box
Box box2;         // Declare variable box2 of type Box
```

Kristen is a contender for valedictorian of her high school. She wants to know how many students (if any) have scored higher than her in the 5 exams given during this semester.

Create a class named $Student$ with the following specifications:

- An instance variable named $scores$ to hold a student's 5 exam scores.
- A void input() function that reads 5 integers and saves them to $scores$.
- An int calculateTotalScore() function that returns the sum of the student's scores.

Input Format

Most of the input is handled for you by the locked code in the editor.

In the `void Student::input()` function, you must read 5 scores from stdin and save them to your $scores$ instance variable.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
3
30 40 45 10 10
40 40 40 10 10
50 20 30 10 10
```

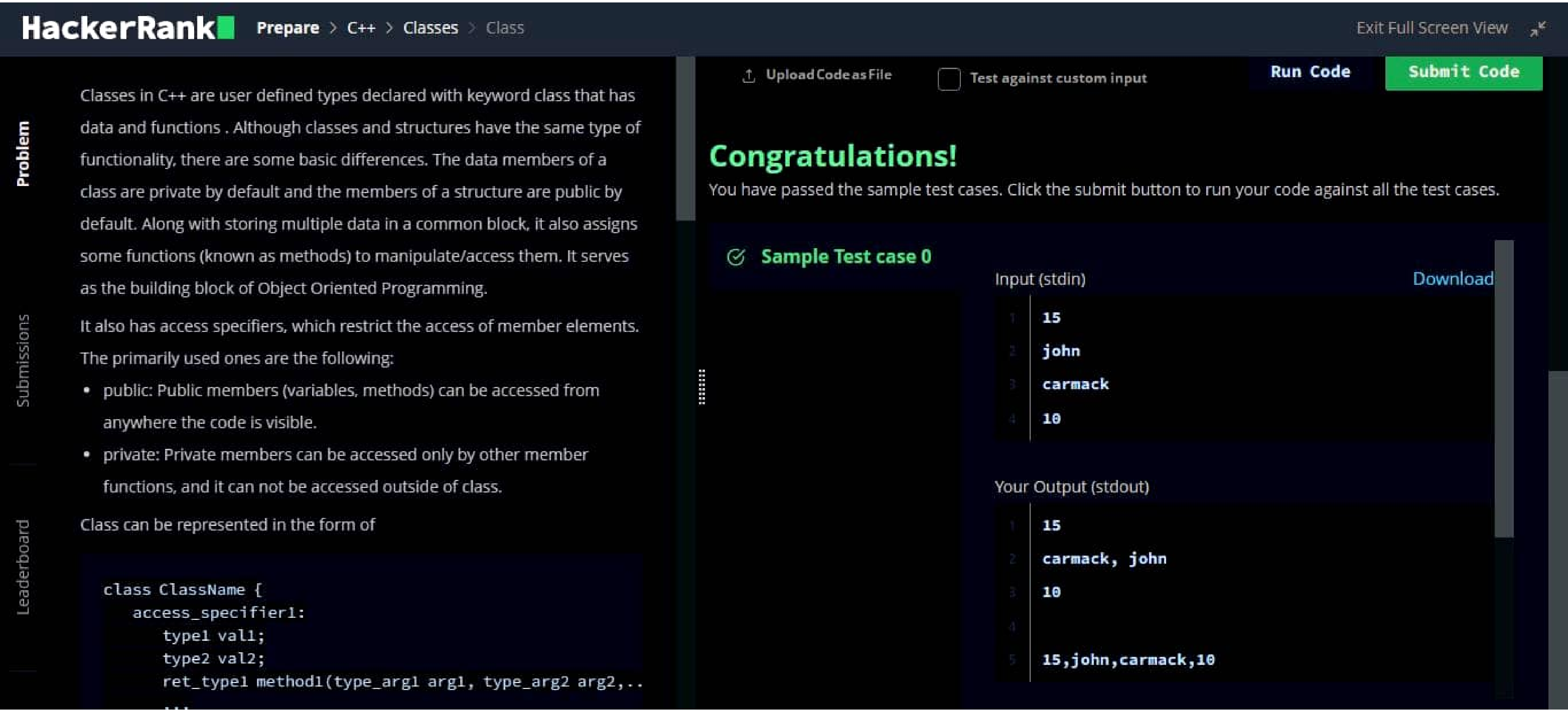
Your Output (stdout)

```
1
```

Expected Output Download

```
1
```

Scanned with CamScanner



Classes in C++ are user defined types declared with keyword class that has data and functions . Although classes and structures have the same type of functionality, there are some basic differences. The data members of a class are private by default and the members of a structure are public by default. Along with storing multiple data in a common block, it also assigns some functions (known as methods) to manipulate/access them. It serves as the building block of Object Oriented Programming.

It also has access specifiers, which restrict the access of member elements. The primarily used ones are the following:

- public: Public members (variables, methods) can be accessed from anywhere the code is visible.
- private: Private members can be accessed only by other member functions, and it can not be accessed outside of class.

Class can be represented in the form of

```
class ClassName {
    access_specifier1:
        type1 val1;
        type2 val2;
        ret_type1 method1(type_arg1 arg1, type_arg2 arg2,..
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

Input (stdin) Download

```
15
john
carmack
10
```

Your Output (stdout)

```
15
carmack, john
10

15,john,carmack,10
```