Prepare > C++ > Introduction > Arrays Introduction

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

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 siz
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 ele
arr[0],the second element at arr[1] and so on thr
```

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

Input Format

The first line of the input contains N,where N is the number of integers.The next line contains N space-separated integers.

Constraints

$1 <= N <= 1000$

$1 <= A[i] <= 10000$, where $A[i]$ is the i^{th} integer in the array.

Output Format

Print the N integers of the array in the reverse order, space-separated on a single line.

Sample Input

```
4
1 4 3 2
```

Sample Output

```
2 3 4 1
```

```
int main() {
    int n;
    cin >> n;
    int arr[n];

    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    for (int i = n - 1; i >= 0; i--) {
        cout << arr[i] << " ";
    }

    return 0;
}
```

Line: 14 Col: 33

Upload Code as File

Run Code

Submit Code

Test against custom input

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
```

Prepare > C++ > Introduction > Basic Data Types   Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions | Editorial

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

To print a data type, use the following syntax:

```
printf("`format_specifier`", val)
```

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

```
char ch = 'd';
double d = 234.432;
printf("%c %lf", ch, d);
```

Note: You can also use cin and cout instead of scanf and printf; however, if you are taking a million numbers as input and printing a million lines, it is faster to use scanf and printf.

Input Format

Input consists of the following space-separated values: int, long, char, float, and double, respectively.

Output Format

Print each element on a new line in the same order it was received as input. Note that the floating point value should be correct up to 3 decimal places and the double to 9 decimal places.

Sample Input

```
3 12345678912345 a 334.23 14049.30493
```

Sample Output

```
3
12345678912345
a
334.230
14049.304930000
```

Explanation

Print int 3,
followed by long 12345678912345,
followed by char a,
followed by float 334.23,
followed by double 14049.30493.

Change Theme   Language C++11

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

int main() {
    int i;
    long l;
    char c;
    float f;
    double d;

    scanf("%d %ld %c %f %lf", &i, &l, &c, &f, &d)

    printf("%d\n", i);
    printf("%ld\n", l);
    printf("%c\n", c);
    printf("%.3f\n", f);
    printf("%.9lf\n", d);

    return 0;
}

```

Line: 22 Col: 1

Upload Code as File   Run Code   Submit Code

Test against custom input

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
334.230
14049.304930000
```

Prepare > C++ > Classes > Box It!

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Design a class named Box whose dimensions are integers and private to the class. The dimensions are labelled: length l, breadth b, and height h.

The default constructor of the class should initialize l, b, and h to 0.

The parameterized constructor Box(int length, int breadth, int height) should initialize Box's l, b and h to length, breadth and height.

The copy constructor Box$(Box\ B)$ should set l, b and h to B's l, b and h, respectively.

Apart from the above, the class should have 4 functions:

- int getLength() - Return box's length
- int getBreadth() - Return box's breadth
- int getHeight() - Return box's height
- long long CalculateVolume() - Return the volume of the box

Overload the operator $<$ for the class Box. Box $A <$ Box B if:

1. $A.l < B.l$
2. $A.b < B.b$ and $A.l == B.l$
3. $A.h < B.h$ and $A.b == B.b$ and $A.l == B.l$

Overload operator $<<$ for the class Box().

If B is an object of class Box:

$cout << B$ should print $B.l$, $B.b$ and $B.h$ on a single line separated by spaces.

For example,

```
Box b1; // Should set b1.l = b1.b = b1.h = 0;
Box b2(2, 3, 4); // Should set b1.l = 2, b1.b = 3, b1.h = 4;
b2.getLength();       // Should return 2
b2.getBreadth(); // Should return 3
b2.getheight();       // Should return 4
b2.CalculateVolume(); // Should return 24
bool x = (b1 < b2);          // Should return true based on t
cout<<b2; // Should print 2 3 4 in order.
```

Constraints

$0 \le l, b, h \le 10^5$

Two boxes being compared using the $<$ operator will not have all three dimensions equal.

Change Theme Language C++11

```cpp
#include<bits/stdc++.h>

using namespace std;
class Box {
private:
    int l, b, h;   // Length, Breadth, Height

public:
    // Default Constructor
    Box() {
        l = 0;
        b = 0;
        h = 0;
    }

    // Parameterized Constructor
    Box(int length, int breadth, int height) {
        l = length;
        b = breadth;
        h = height;
    }

    // Copy Constructor
    Box(const Box &B) {
        l = B.l;
        b = B.b;
        h = B.h;
    }

    // Getters
    int getLength() const { return l; }
    int getBreadth() const { return b; }
    int getHeight() const { return h; }

    // Calculate Volume
```

Line: 60 Col: 1

Upload Code as File   Run Code   Submit Code

Test against custom input

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

```
5
2 3 4 5
4
5
4
2 4 6 7
```

Your Output (stdout)

```
3 4 5
60
```

Prepare > C++ > Classes > Class

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

```
        void set(int a) {
            val = a;
        }
        int get() {
            return val;
        }
    };
```

We can store details related to a student in a class consisting of his age (int), first_name (string), last_name (string) and standard (int).

You have to create a class, named Student, representing the student's details, as mentioned above, and store the data of a student. Create setter and getter functions for each element; that is, the class should at least have following functions:

- get_age, set_age
- get_first_name, set_first_name
- get_last_name, set_last_name
- get_standard, set_standard

Also, you have to create another method to_string() which returns the string consisting of the above elements, separated by a comma(,). You can refer to stringstream for this.

Input Format

Input will consist of four lines.
The first line will contain an integer, representing the age. The second line will contain a string, consisting of lower-case Latin characters ('a'-'z'), representing the first_name of a student.
The third line will contain another string, consisting of lower-case Latin characters ('a'-'z'), representing the last_name of a student.
The fourth line will contain an integer, representing the standard of student.

Note: The number of characters in first_name and last_name will not exceed 50.

Output Format

The code provided by HackerRank will use your class members to set and then get the elements of the Student class.

Sample Input

```
15
john
carmack
10
```

Sample Output

```
15
carmack, john
10

15,john,carmack,10
```

Change Theme Language C++11

```
class Student {void set_standard(int s) {
    }

    int get_standard() {
        return standard;
    }

    string to_string() {
        stringstream ss;
        ss << age << "," << first_name << "," <<
        return ss.str();
    }
};

int main() {
    int age, standard;
    string first_name, last_name;

    cin >> age >> first_name >> last_name >> stan

    Student st;
    st.set_age(age);
    st.set_standard(standard);
    st.set_first_name(first_name);
    st.set_last_name(last_name);

    cout << st.get_age() << "\n";
    cout << st.get_last_name() << ", " << st.get_
    cout << st.get_standard() << "\n";
    cout << "\n";
    cout << st.to_string();

    return 0;
}
```

Line: 72 Col: 1

Upload Code as File Run Code Submit Code

Test against custom input

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

```

Prepare > C++ > Classes > Classes and Objects   Exit Full Screen View

Problem

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 $\boldsymbol{Student}$ with the following specifications:

- An instance variable named $\boldsymbol{scores}$ to hold a student's 5 exam scores.
- A void input() function that reads 5 integers and saves them to $\boldsymbol{scores}$.
- 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 $\boldsymbol{scores}$ instance variable.

Constraints

$1 \le n \le 100$

$0 \le examscore \le 50$

Output Format

In the `int Student::calculateTotalScore()` function, you must return the student's total grade (the sum of the values in $\boldsymbol{scores}$).

The locked code in the editor will determine how many scores are larger than Kristen's and print that number to the console.

Sample Input

The first line contains n, the number of students in Kristen's class. The n subsequent lines contain each student's 5 exam grades for this semester.

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

Sample Output

```
1
```

Explanation

Kristen's grades are on the first line of grades. Only 1 student scored higher than her.

Submissions

Leaderboard

Discussions

```cpp
#include <vector>
#include <iostream>
#include <algorithm>
#include <cassert>
using namespace std;
class Student {
private:
    vector<int> scores; // to store the 5 exam scores

public:
    void input() {
        // Read 5 scores from input
        for(int i = 0; i < 5; i++) {
            int score;
            cin >> score;
            scores.push_back(score);
        }
    }

    int calculateTotalScore() {
        int total = 0;
        for(int i = 0; i < scores.size(); i++) {
            total += scores[i];
        }
        return total;
    }
};


int main() {
    int n; // number of students
    cin >> n;
    Student *s = new Student[n]; // an array of n students
```

Line: 30 Col: 1

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

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

Your Output (stdout)

```
1
```

Expected Output   Download

```
1
```

Prepare > C++ > Introduction > Conditional Statements

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Editorial

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 {
   statement2;
   ...
}
```

3. if - else if - else: In this structure, dependent statements are chained together and the $condition$ for each statement is only checked if all prior conditions in the chain evaluated to false. Once a $condition$ evaluates to true, the bracketed code associated with that statement is executed and the program then skips to the end of the chain of statements and continues executing. If each $condition$ in the chain evaluates to false, then the body of bracketed code in the else block at the end is executed.

```
if(first condition) {
   ...
}
else if(second condition) {
   ...
}
.
.
.
else if((n-1)'th condition) {
   ....
}
else {
   ...
}
```

Given a positive integer n, do the following:

- If $1 \leq n \leq 9$, print the lowercase English word corresponding to the number (e.g., one for 1, two for 2, etc.).
- If $n > 9$, print `Greater than 9`.

Change Theme   Language   C++11

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

string ltrim(const string &);
string rtrim(const string &);

int main()
{
    string n_temp;
    getline(cin, n_temp);

    int n = stoi(ltrim(rtrim(n_temp)));

    if (n == 1)
        cout << "one";
    else if (n == 2)
        cout << "two";
    else if (n == 3)
        cout << "three";
    else if (n == 4)
        cout << "four";
    else if (n == 5)
        cout << "five";
    else if (n == 6)
        cout << "six";
    else if (n == 7)
        cout << "seven";
    else if (n == 8)
        cout << "eight";
    else if (n == 9)
        cout << "nine";
    else
        cout << "Greater than 9";

    return 0;
```

Line: 37 Col: 1

Upload Code as File   Run Code   Submit Code

Test against custom input

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
```

Prepare > C++ > Classes > Exceptional Server Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

In this challenge, you are required to handle error messages while working with small computational server that performs complex calculations.

It has a function that takes **2** large numbers as its input and returns a numeric result. Unfortunately, there are various exceptions that may occur during execution.

Complete the code in your editor so that it prints appropriate error messages, should anything go wrong. The expected behavior is defined as follows:

- If the compute function runs fine with the given arguments, then print the result of the function call.
- If it fails to allocate the memory that it needs, print `Not enough memory`.
- If any other standard C++ exception occurs, print `Exception: S` where S is the exception's error message.
- If any non-standard exception occurs, print `Other Exception`.

Input Format

The first line contains an integer, T, the number of test cases.

Each of the T subsequent lines describes a test case as **2** space-separated integers, A and B, respectively.

Constraints

$1 \leq T \leq 10^3$

$0 \leq A, B \leq 2^{60}$

Output Format

For each test case, print a single line containing whichever message described in the Problem Statement above is appropriate. After all messages have been printed, the locked stub code in your editor prints the server load.

Sample Input

```
2
-8 5
1435434255433 5
```

Sample Output

```
Exception: A is negative
Not enough memory
2
```

Explanation

-8 is negative, hence 'Exception: A is negative' is thrown. Since the second input is too large, 'not enough memory' is displayed. **2** is the server load.

Change Theme Language C++11

```
class Server {
        int real = -1, cmplx = sqrt(-1);
    static int compute(long long A, long long B)
        if(B == 0) throw 0;
        real = (A/B)*real;
        int ans = v.at(B);
        return real + A - B*ans;
    }
    static int getLoad() {
        return load;
    }
};
int Server::load = 0;

int main() {
    int T; cin >> T;
    while(T--) {
        long long A, B;
        cin >> A >> B;

    try {
            cout << Server::compute(A, B) << endl
        }
        catch (bad_alloc& e) {
            cout << "Not enough memory" << endl;
        }
        catch (exception& e) {
            cout << "Exception: " << e.what() <<
        }
        catch (...) {
            cout << "Other Exception" << endl;
        }

    }
    cout << Server::getLoad() << endl;
    return 0;
}
```

Line: 48 Col: 10

Upload Code as File Run Code Submit Code

Test against custom input

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
2 -8 5
3 1435434255433 5
```

Your Output (stdout)

```
1 Exception: A is
  negative
2 Not enough memory
3 2
```

Prepare > C++ > Introduction > For Loop   Exit Full Screen View

Problem

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

The syntax is

```
for ( <expression_1> ; <expression_2> ; <express
    <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.

Input Format

You will be given two positive integers, a and b ($a \leq b$), separated by a newline.

Output Format

For each integer n in the inclusive interval $[a, b]$:

- If $1 \leq n \leq 9$, then print the English representation of it in lowercase. That is "one" for 1, "two" for 2, and so on.
- Else if $n > 9$ and it is an even number, then print "even".
- Else if $n > 9$ and it is an odd number, then print "odd".

Note: $[a, b] = \{x \in \mathbb{Z} \mid a \leq x \leq b\} = \{a,\ a+1, \ldots, b\}$

Sample Input

```
8
11
```

Sample Output

```
eight
nine
even
odd
```

Submissions Leaderboard Discussions Editorial

Change Theme   Language C++11

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

int main() {
    int a, b;
    cin >> a >> b;

    for (int i = a; i <= b; i++) {
        if (i == 1)
            cout << "one" << endl;
        else if (i == 2)
            cout << "two" << endl;
        else if (i == 3)
            cout << "three" << endl;
        else if (i == 4)
            cout << "four" << endl;
        else if (i == 5)
            cout << "five" << endl;
        else if (i == 6)
            cout << "six" << endl;
        else if (i == 7)
            cout << "seven" << endl;
        else if (i == 8)
            cout << "eight" << endl;
        else if (i == 9)
            cout << "nine" << endl;
        else if (i > 9 && i % 2 == 0)
            cout << "even" << endl;
        else
            cout << "odd" << endl;
    }

    return 0;
}
```

Line: 6 Col: 20

Upload Code as File   Run Code   Submit Code

Test against custom input

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
```

Prepare > C++ > Introduction > Functions

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Editorial

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_
    ...
    ...
    ...
    [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.

```
+= : Add and assignment operator. It adds the rig
a += b is equivalent to a = a + b;
```

Input Format

Input will contain four integers - a, b, c, d , one per line.

Output Format

Return the greatest of the four integers.

PS: I/O will be automatically handled.

Sample Input

```
3
4
6
5
```

Sample Output

```
6
```

Change Theme Language C++11

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

int max_of_four(int a, int b, int c, int d) {
    int max = a;
    if (b > max) {
        max = b;
    }
    if (c > max) {
        max = c;
    }
    if (d > max) {
        max = d;
    }
    return max;
}

int main() {
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    int result = max_of_four(a, b, c, d);
    cout << result;
    return 0;
}

```

Line: 25 Col: 1

Upload Code as File Run Code Submit Code

Test against custom input

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  3
2  4
3  6
4  5
```

Your Output (stdout)

```
1  6
```

Expected Output Download

Prepare > C++ > Classes > Inherited Code Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Editorial

You inherited a piece of code that performs username validation for your company's website. The existing function works reasonably well, but it throws an exception when the username is too short. Upon review, you realize that nobody ever defined the exception.

The inherited code is provided for you in the locked section of your editor. Complete the code so that, when an exception is thrown, it prints `Too short: n` (where n is the length of the given username).

Input Format

The first line contains an integer, t, the number of test cases.

Each of the t subsequent lines describes a test case as a single username string, u.

Constraints

- $1 \le t \le 1000$
- $1 \le |u| \le 100$
- The username consists only of uppercase and lowercase letters.

Output Format

You are not responsible for directly printing anything to stdout. If your code is correct, the locked stub code in your editor will print either `Valid` (if the username is valid), `Invalid` (if the username is invalid), or `Too short: n` (where n is the length of the too-short username) on a new line for each test case.

Sample Input

```
3
Peter
Me
Arxwwz
```

Sample Output

```
Valid
Too short: 2
Invalid
```

Explanation

Username `Me` is too short because it only contains 2 characters, so your exception prints **`Too short: 2`**.

All other validation is handled by the locked code in your editor.

Change Theme Language C++11

```
#include <iostream>
#include <string>
#include <sstream>
#include <exception>
using namespace std;

class BadLengthException : public exception {
private:
    int length;
    string msg;
public:
    BadLengthException(int n) {
        length = n;
        msg = to_string(n);   // store result so
    }
    const char* what() const noexcept {
        return msg.c_str();
    }
};



bool checkUsername(string username) {
    bool isValid = true;
    int n = username.length();
    if(n < 5) {
        throw BadLengthException(n);
    }
    for(int i = 0; i < n-1; i++) {
        if(username[i] == 'w' && username[i+1] ==
            isValid = false;
        }
    }
    return isValid;
}
```

Line: 21 Col: 1

Upload Code as File Run Code Submit Code

Test against custom input

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
Peter
Me
Arxwwz
```

Your Output (stdout)

```
Valid
Too short: 2
Invalid
```

Prepare > C++ > Introduction > Input and Output Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Editorial

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:

```
cout << s << " " << n << endl;
```

This code prints the contents of string s, a single space ($" "$), then the integer n. We end our line of output with a newline using endl. This results in the following output:

```
High 5
```

Task

Read 3 numbers from stdin and print their sum to stdout.

Input Format

One line that contains 3 space-separated integers: a, b, and c.

Constraints

- $1 \leq a, b, c \leq 1000$

Output Format

Print the sum of the three numbers on a single line.

Sample Input

```
1 2 7
```

Sample Output

```
10
```

Explanation

The sum of the three numbers is $1 + 2 + 7 = 10$.

Change Theme Language C++11

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
   int a, b, c;
      cin >> a >> b >> c;
      cout << a + b + c <<endl;
    return 0;
}

```

Line: 15 Col: 1

Upload Code as File Run Code Submit Code

Test against custom input

You have earned 5.00 points!

You are now 5 points away from the 1st star for your c++ badge.

50% 5/10

Congratulations

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

Next Challenge

Test case 0 Compiler Message

Prepare > C++ > Introduction > Input and Output

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Editorial

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:

```
cout << s << " " << n << endl;
```

This code prints the contents of string s, a single space (" "), then the integer n. We end our line of output with a newline using endl. This results in the following output:

```
High 5
```

Task

Read 3 numbers from stdin and print their sum to stdout.

Input Format

One line that contains 3 space-separated integers: a, b, and c.

Constraints

- $1 \leq a, b, c \leq 1000$

Output Format

Print the sum of the three numbers on a single line.

Sample Input

```
1 2 7
```

Sample Output

```
10
```

Explanation

The sum of the three numbers is $1 + 2 + 7 = 10$.

Change Theme Language C++11

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
   int a, b, c;
      cin >> a >> b >> c;
      cout << a + b + c <<endl;
    return 0;
}

```

Line: 15 Col: 1

Upload Code as File

Run Code

Submit Code

Test against custom input

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

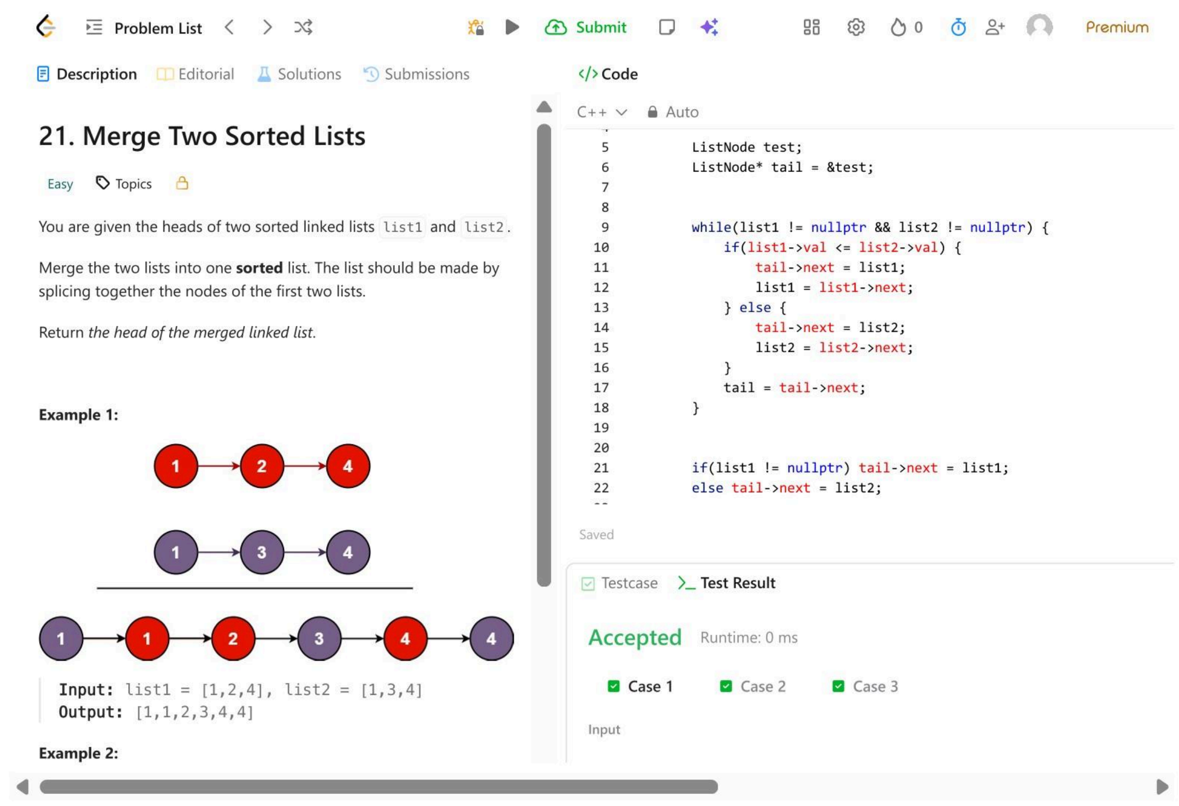
```
1 2 7
```

Your Output (stdout)

```
10
```

Expected Output Download

```
10
```



21. Merge Two Sorted Lists

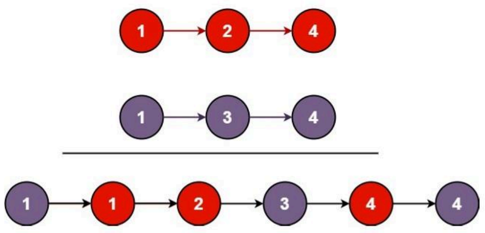
Easy Topics

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list*.

Example 1:



```
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]
```

Example 2:

```
        ListNode test;
        ListNode* tail = &test;


        while(list1 != nullptr && list2 != nullptr) {
            if(list1->val <= list2->val) {
                tail->next = list1;
                list1 = list1->next;
            } else {
                tail->next = list2;
                list2 = list2->next;
            }
            tail = tail->next;
        }


        if(list1 != nullptr) tail->next = list1;
        else tail->next = list2;
```

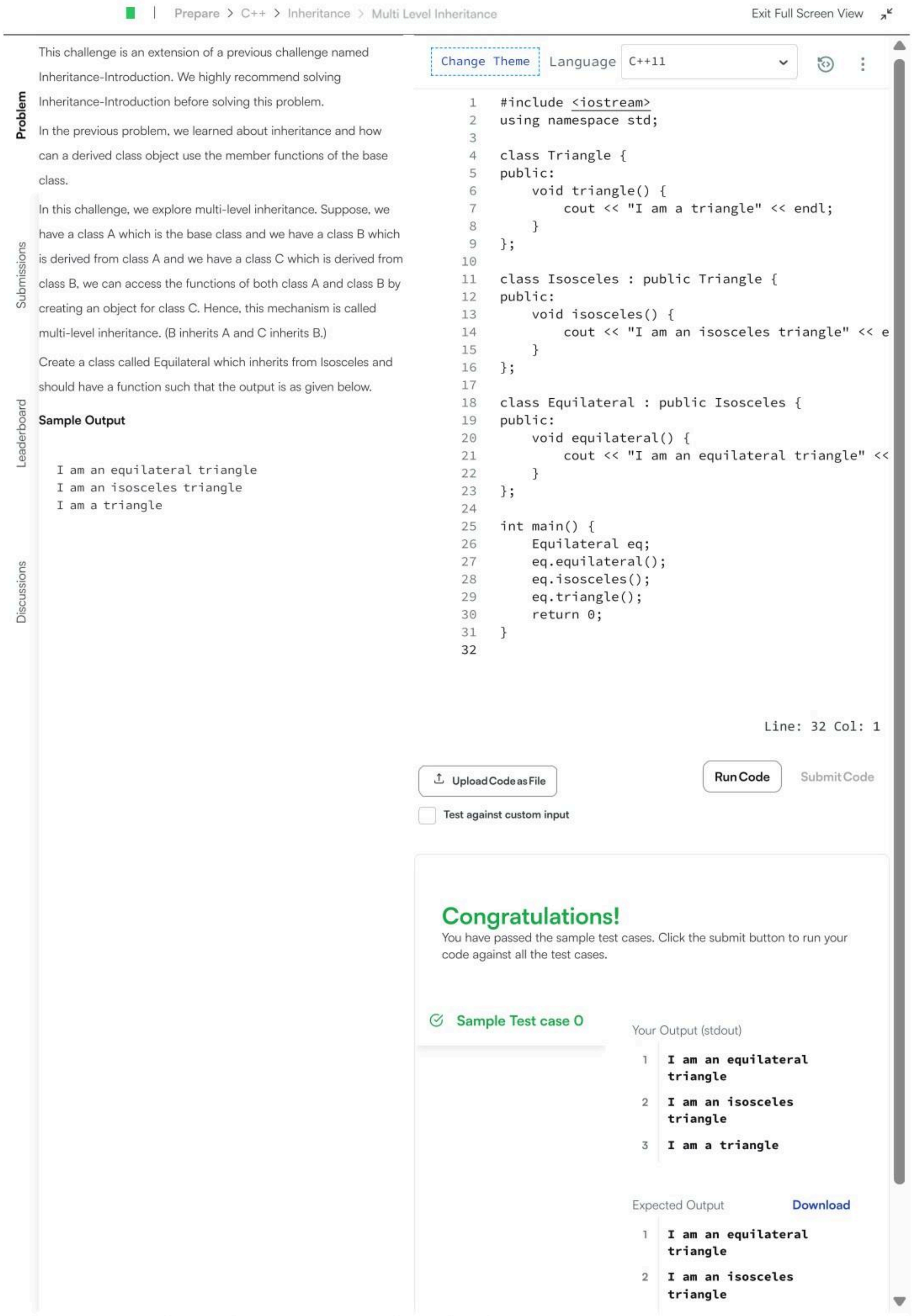
Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input



Prepare > C++ > Inheritance > Multi Level Inheritance

This challenge is an extension of a previous challenge named Inheritance-Introduction. We highly recommend solving Inheritance-Introduction before solving this problem.

In the previous problem, we learned about inheritance and how can a derived class object use the member functions of the base class.

In this challenge, we explore multi-level inheritance. Suppose, we have a class A which is the base class and we have a class B which is derived from class A and we have a class C which is derived from class B, we can access the functions of both class A and class B by creating an object for class C. Hence, this mechanism is called multi-level inheritance. (B inherits A and C inherits B.)

Create a class called Equilateral which inherits from Isosceles and should have a function such that the output is as given below.

Sample Output

```
I am an equilateral triangle
I am an isosceles triangle
I am a triangle
```

Language C++11

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

class Triangle {
public:
    void triangle() {
        cout << "I am a triangle" << endl;
    }
};

class Isosceles : public Triangle {
public:
    void isosceles() {
        cout << "I am an isosceles triangle" << e
    }
};

class Equilateral : public Isosceles {
public:
    void equilateral() {
        cout << "I am an equilateral triangle" <<
    }
};

int main() {
    Equilateral eq;
    eq.equilateral();
    eq.isosceles();
    eq.triangle();
    return 0;
}
```

Line: 32 Col: 1

Congratulations!

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

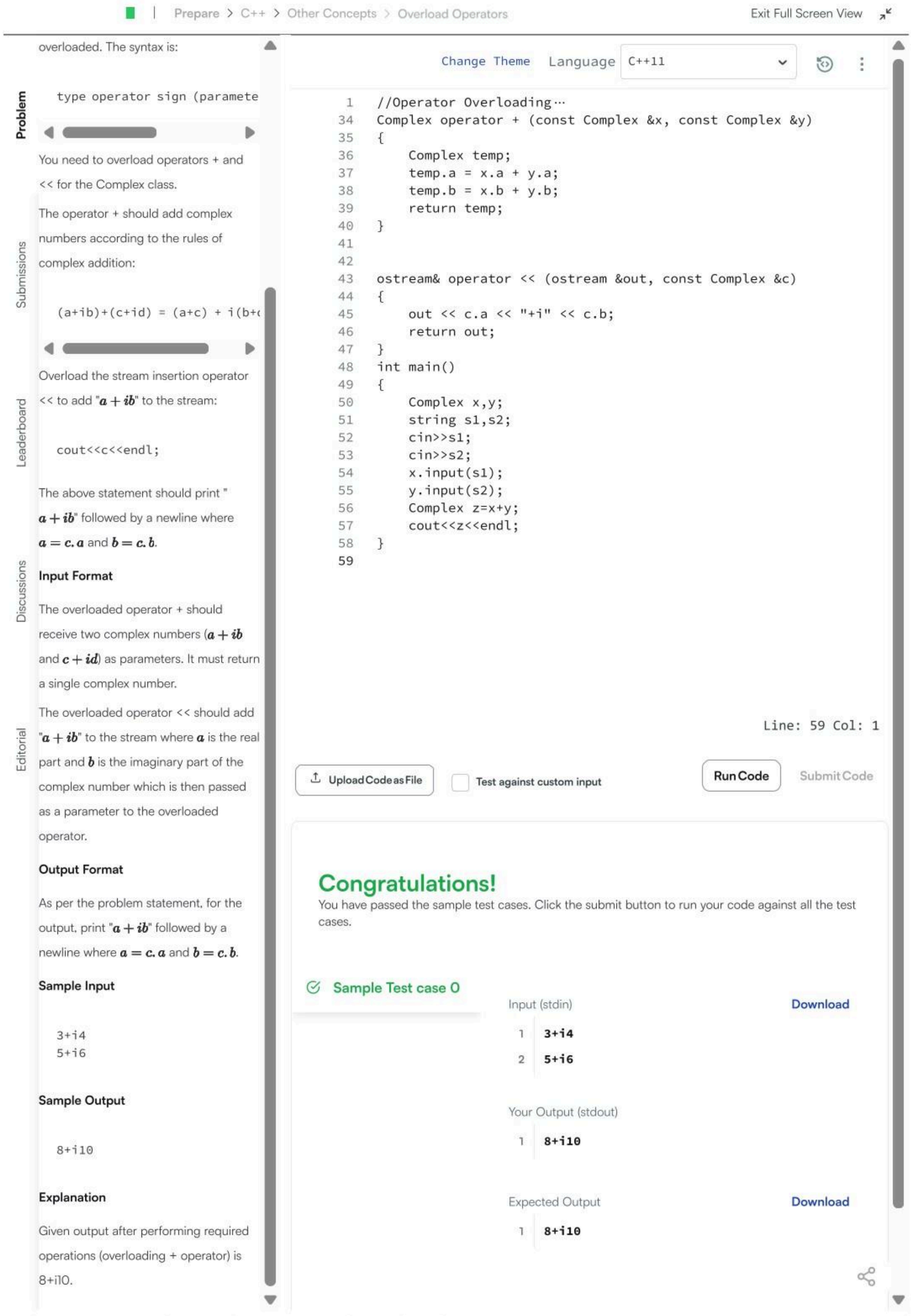
Sample Test case 0

Your Output (stdout)

```
I am an equilateral triangle
I am an isosceles triangle
I am a triangle
```

Expected Output

```
I am an equilateral triangle
I am an isosceles triangle
```



Prepare > C++ > Other Concepts > Overload Operators   Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions | Editorial

overloaded. The syntax is:

```
type operator sign (paramete
```

You need to overload operators + and << for the Complex class.

The operator + should add complex numbers according to the rules of complex addition:

```
(a+ib)+(c+id) = (a+c) + i(b+
```

Overload the stream insertion operator << to add "$a + ib$" to the stream:

```
cout<<c<<endl;
```

The above statement should print "$a + ib$" followed by a newline where $a = c.a$ and $b = c.b$.

Input Format

The overloaded operator + should receive two complex numbers ($a + ib$ and $c + id$) as parameters. It must return a single complex number.

The overloaded operator << should add "$a + ib$" to the stream where a is the real part and b is the imaginary part of the complex number which is then passed as a parameter to the overloaded operator.

Output Format

As per the problem statement, for the output, print "$a + ib$" followed by a newline where $a = c.a$ and $b = c.b$.

Sample Input

```
3+i4
5+i6
```

Sample Output

```
8+i10
```

Explanation

Given output after performing required operations (overloading + operator) is 8+i10.

Change Theme   Language C++11

```
//Operator Overloading···
Complex operator + (const Complex &x, const Complex &y)
{
    Complex temp;
    temp.a = x.a + y.a;
    temp.b = x.b + y.b;
    return temp;
}


ostream& operator << (ostream &out, const Complex &c)
{
    out << c.a << "+i" << c.b;
    return out;
}
int main()
{
    Complex x,y;
    string s1,s2;
    cin>>s1;
    cin>>s2;
    x.input(s1);
    y.input(s2);
    Complex z=x+y;
    cout<<z<<endl;
}

```

Line: 59 Col: 1

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

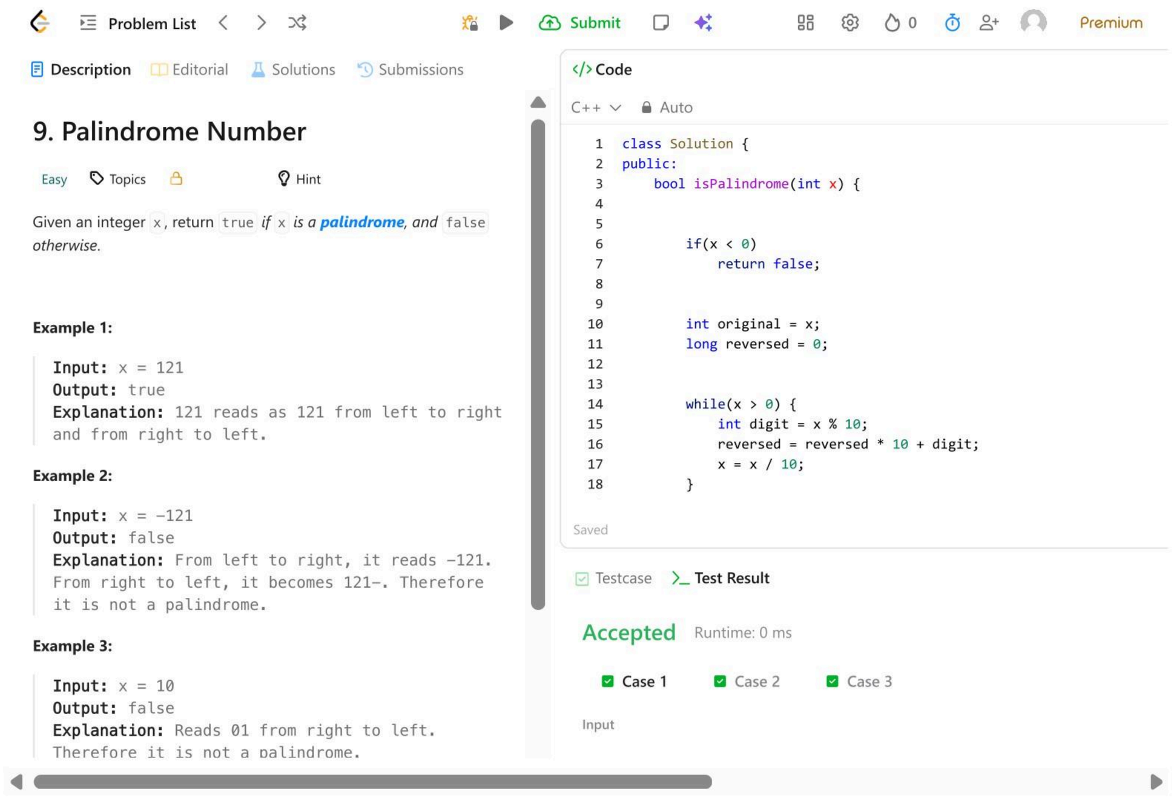
```
3+i4
5+i6
```

Your Output (stdout)

```
8+i10
```

Expected Output   Download

```
8+i10
```



9. Palindrome Number

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Example 1:

```
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right
and from right to left.
```

Example 2:

```
Input: x = -121
Output: false
Explanation: From left to right, it reads -121.
From right to left, it becomes 121-. Therefore
it is not a palindrome.
```

Example 3:

```
Input: x = 10
Output: false
Explanation: Reads 01 from right to left.
Therefore it is not a palindrome.
```

```cpp
class Solution {
public:
    bool isPalindrome(int x) {


        if(x < 0)
            return false;


        int original = x;
        long reversed = 0;


        while(x > 0) {
            int digit = x % 10;
            reversed = reversed * 10 + digit;
            x = x / 10;
        }
```

Accepted Runtime: 0 ms

✓ Case 1 ✓ Case 2 ✓ Case 3

Input

Prepare > C++ > Introduction > Pointer

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Editorial

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);
   return 0;
}
```

Function Description

Complete the update function in the editor below.

update has the following parameters:

- int *a: an integer
- int *b: an integer

Returns

- The function is declared with a `void` return type, so there is no value to return. Modify the values in memory so that a contains their sum and b contains their absoluted difference.
- $a' = a + b$
- $b' = |a - b|$

Input Format

Input will contain two integers, a and b, separated by a newline.

Sample Input

```
4
5
```

Sample Output

```
9
1
```

Explanation

- $a' = 4 + 5 = 9$
- $b' = |4 - 5| = 1$

Change Theme Language C++11

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

void update(int *a, int *b) {
    int sum = *a + *b;
    int diff = *a - *b;
    if (diff < 0) {
        diff = -diff;
    }
    *a = sum;
    *b = diff;
}

int main() {
    int a, b;
    cin >> a >> b;
    update(&a, &b);
    cout << a << endl;
    cout << b << endl;
    return 0;
}
```

Line: 22 Col: 1

Upload Code as File Run Code Submit Code

Test against custom input

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
```

Prepare > C++ > Inheritance > Rectangle Area Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

In this challenge, you are required to compute the area of a rectangle using classes.

Create two classes:

Rectangle

The Rectangle class should have two data fields-width and height of int types. The class should have display() method, to print the width and height of the rectangle separated by space.

RectangleArea

The RectangleArea class is derived from Rectangle class, i.e., it is the sub-class of Rectangle class. The class should have read_input() method, to read the values of width and height of the rectangle. The RectangleArea class should also overload the display() method to print the area $(\mathbf{width} \times \mathbf{height})$ of the rectangle.

Input Format

The first and only line of input contains two space separated integers denoting the width and height of the rectangle.

Constraints

- $1 \leq width, height \leq 100$

Output Format

The output should consist of exactly two lines:
In the first line, print the width and height of the rectangle separated by space.
In the second line, print the area of the rectangle.

Sample Input

```
10 5
```

Sample Output

```
10 5
50
```

Explanation

As, $width = 10$ and $height = 5$, so
$area = width \times height = 50$

```
class Rectangle {
protected:
    int width, height;
public:
    void display() {
        cout << width << " " << height << endl;
    }
};

class RectangleArea : public Rectangle {
public:
    void read_input() {
        cin >> width >> height;
    }

    void display() {
        cout << width * height << endl;
    }
};

int main()
{
    /*
     * Declare a RectangleArea object
     */
    RectangleArea r_area;

    /*
     * Read the width and height
     */
    r_area.read_input();

```

Line: 22 Col: 3

Upload Code as File Run Code Submit Code

Test against custom input

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 10 5
```

Your Output (stdout)

```
1 10 5
2 50
```

Expected Output Download

```
1 10 5
2 50
```

Problem List Submit Premium

Description Editorial Solutions Submissions

917. Reverse Only Letters

Easy Topics Hint

Given a string `s`, reverse the string according to the following rules:

- All the characters that are not English letters remain in the same position.
- All the English letters (lowercase or uppercase) should be reversed.

Return `s` *after reversing it.*

Example 1:

```
Input: s = "ab-cd"
Output: "dc-ba"
```

Example 2:

```
Input: s = "a-bC-dEf-ghIj"
Output: "j-Ih-gfE-dCba"
```

Example 3:

```
Input: s = "Test1ng-Leet=code-Q!"
Output: "Qedo1ct-eeLg=ntse-T!"
```

Constraints:

- `1 <= s.length <= 100`
- `s` consists of characters with ASCII values in the range `[33, 122]`.
- `s` does not contain `'\"'` or `'\\'`.

2.4K 40 7 Online

Code

C++ Auto

```cpp
class Solution {
public:
    string reverseOnlyLetters(string s) {
        int left = 0;
        int right = s.size() - 1;

        while (left < right) {
            if (!isalpha(s[left])) {
                left++;
            }
            else if (!isalpha(s[right])) {
                right--;
            }
            else {
                swap(s[left], s[right]);
                left++;
                right--;
            }
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

✓ Case 1 ✓ Case 2 ✓ Case 3

Input

s =

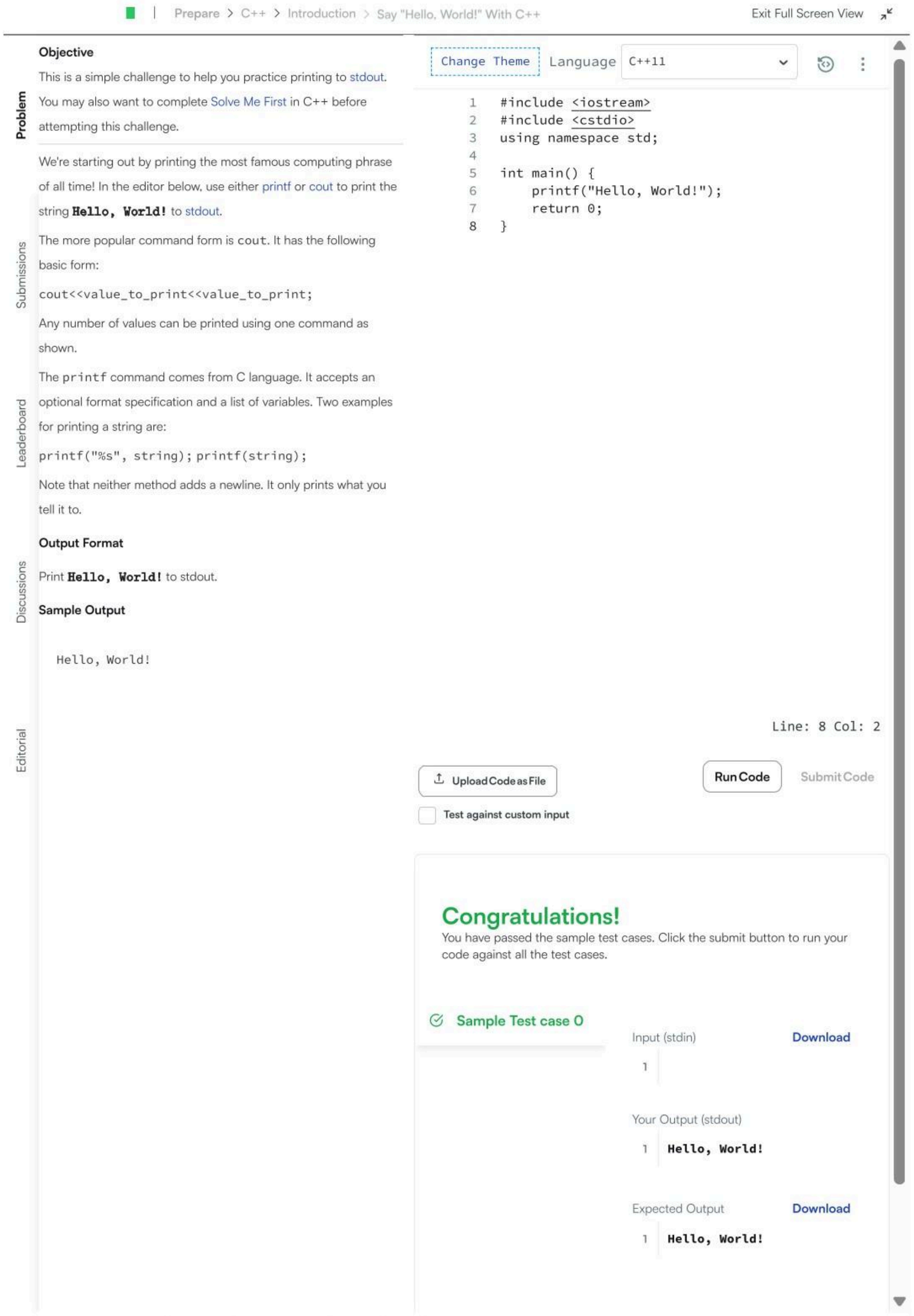
```
"ab-cd"
```

Output

```
"dc-ba"
```

Expected

```
"dc-ba"
```



Prepare > C++ > Introduction > Say "Hello, World!" With C++ Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Editorial

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.

Sample Output

```
Hello, World!
```

Change Theme Language C++11

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

int main() {
    printf("Hello, World!");
    return 0;
}
```

Line: 8 Col: 2

Upload Code as File Run Code Submit Code

Test against custom input

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
```

Your Output (stdout)

```
1  Hello, World!
```

Expected Output Download

```
1  Hello, World!
```

Problem Submissions Leaderboard Discussions

```
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'
char   c1 = s[1];   // c1 = 'b'
char   c2 = s[2];   // c2 = 'c'

s[0] = 'z';        // s = "zbc"
```

P.S.: We will use cin/cout to read/write a string.

Input Format

You are given two strings, a and b, separated by a new line. Each string will consist of lower case Latin characters ('a'-'z').

Output Format

In the first line print two space-separated integers, representing the length of a and b respectively.
In the second line print the string produced by concatenating a and b ($a + b$).
In the third line print two strings separated by a space, a' and b'. a' and b' are the same as a and b, respectively, except that their first characters are swapped.

Sample Input

```
abcd
ef
```

Sample Output

```
4 2
abcdef
ebcd af
```

Explanation

- $a =$ "abcd"
- $b =$ "ef"
- $|a| = 4$
- $|b| = 2$
- $a + b =$ "abcdef"
- $a' =$ "ebcd"
- $b' =$ "af"

```

int main() {
    string a, b;
    cin >> a >> b;

    cout << a.size() << " " << b.size() << endl;

    cout << a + b << endl;

    char temp = a[0];
    a[0] = b[0];
    b[0] = temp;

    cout << a << " " << b << endl;

    return 0;
}

```

Line: 21 Col: 1

Upload Code as File Run Code Submit Code

Test against custom input

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

```
abcd
ef
```

Your Output (stdout)

```
4 2
abcdef
ebcd af
```

Expected Output Download

```
4 2
abcdef
ebcd af
```

Prepare > C++ > Strings > StringStream

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

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
```

Here ch is a storage area for the discarded commas.

If the >> operator returns a value, that is a true value for a conditional. Failure to return a value is false.

Given a string of comma delimited integers, return a vector of integers.

Function Description

Complete the parseInts function in the editor below.

parseInts has the following parameters:

- string str: a string of comma separated integers

Returns

- vector<int>: a vector of the parsed integers.

Note You can learn to push elements onto a vector by solving the first problem in the STL chapter.

Input Format

There is one line of n integers separated by commas.

Constraints

The length of str is less than 8×10^5.

Sample Input

```
23,4,56
```

Sample Output

```
23
4
56
```

```
using namespace std;

vector<int> parseInts(string str) {
    stringstream ss(str);
    vector<int> result;
    int num;
    char ch;

    while (ss >> num) {
        result.push_back(num);
        ss >> ch;
    }

    return result;
}

int main() {
    string str;
    cin >> str;

    vector<int> integers = parseInts(str);
    for (int i = 0; i < integers.size(); i++) {
        cout << integers[i] << "\n";
    }

    return 0;
}
```

Line: 12 Col: 26

Upload Code as File

Run Code

Submit Code

Test against custom input

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
```

Prepare > C++ > Classes > Structs

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

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.
The first line will contain an integer, representing age.
The second line will contain a string, consisting of lower-case Latin characters ('a'-'z'), representing the first_name of a student.
The third line will contain another string, consisting of lower-case Latin characters ('a'-'z'), representing the last_name of a student.
The fourth line will contain an integer, representing the standard of student.

Note: The number of characters in first_name and last_name will not exceed 50.

Output Format

Output will be of a single line, consisting of age, first_name, last_name and standard, each separated by one white space.

P.S.: I/O will be handled by HackerRank.

Sample Input

```
15
john
carmack
10
```

Sample Output

```
15 john carmack 10
```

```
struct Student {
    int age;
    string first_name;
    string last_name;
    int standard;
};

int main() {
    Student st;

    cin >> st.age >> st.first_name >> st.last_nam
    cout << st.age << " " << st.first_name << " "

    return 0;
}
```

Line: 20 Col: 1

Upload Code as File

Run Code

Submit Code

Test against custom input

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

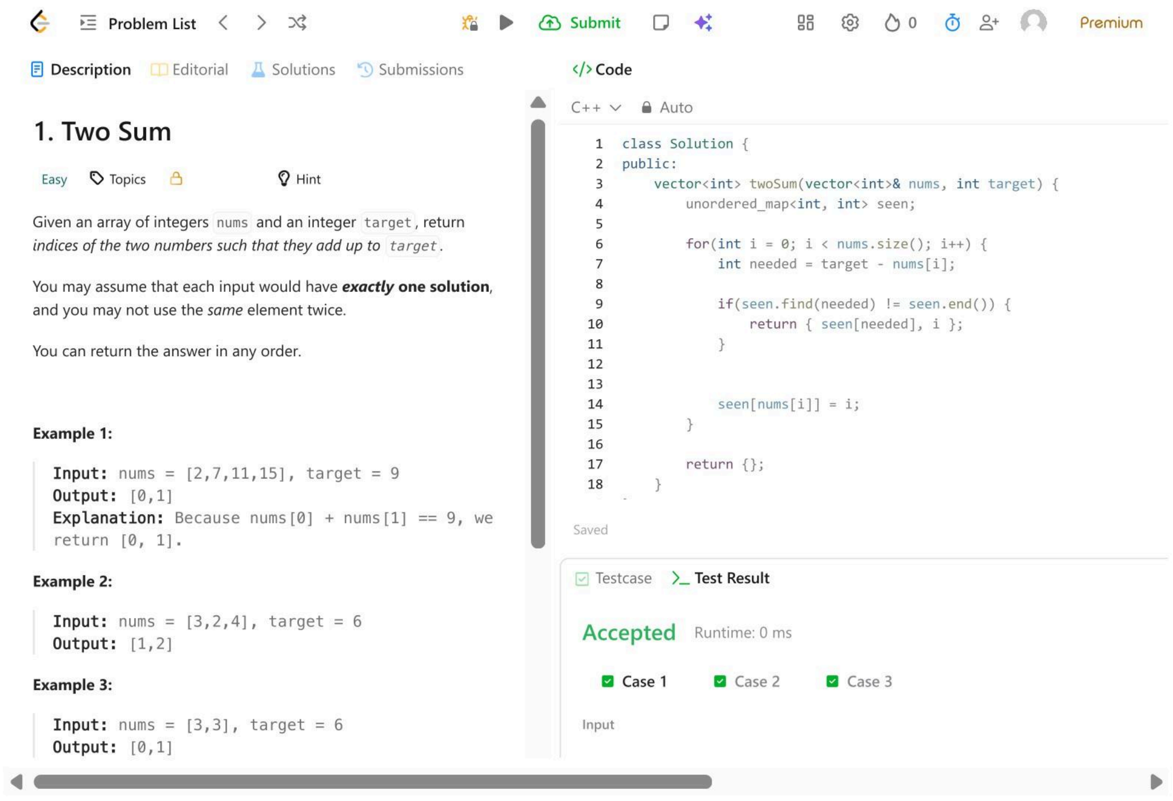
```
15
john
carmack
10
```

Your Output (stdout)

```
15 john carmack 10
```

Expected Output Download

```
15 john carmack 10
```



1. Two Sum

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to `target`*.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we
return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

C++ Auto

```cpp
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> seen;

        for(int i = 0; i < nums.size(); i++) {
            int needed = target - nums[i];

            if(seen.find(needed) != seen.end()) {
                return { seen[needed], i };
            }


            seen[nums[i]] = i;
        }

        return {};
    }
```

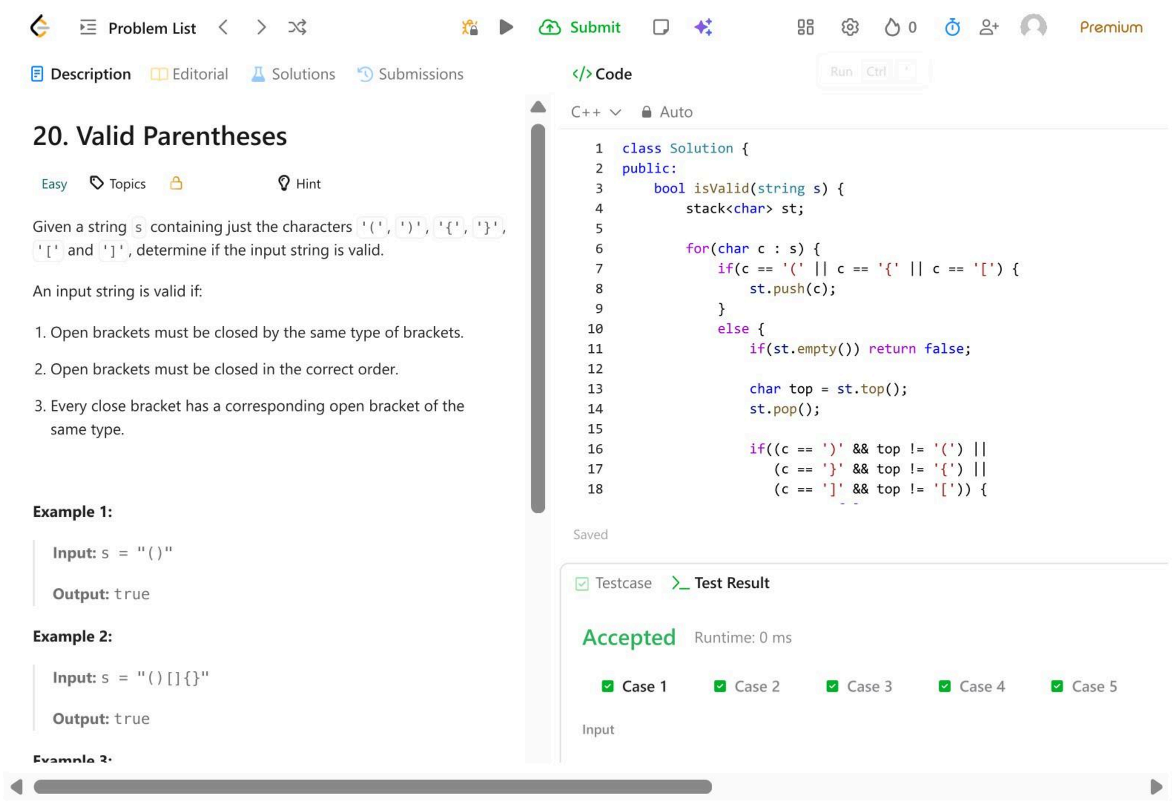
Saved

Testcase Test Result

Accepted Runtime: 0 ms

✓ Case 1 ✓ Case 2 ✓ Case 3

Input



Problem List Submit 0 Premium

Description Editorial Solutions Submissions

Code

20. Valid Parentheses

Easy Topics Hint

Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

Example 1:

Input: `s = "()"`

Output: `true`

Example 2:

Input: `s = "()[]{}"`

Output: `true`

Example 3:

C++ Auto

```cpp
class Solution {
public:
    bool isValid(string s) {
        stack<char> st;

        for(char c : s) {
            if(c == '(' || c == '{' || c == '[') {
                st.push(c);
            }
            else {
                if(st.empty()) return false;

                char top = st.top();
                st.pop();

                if((c == ')' && top != '(') ||
                   (c == '}' && top != '{') ||
                   (c == ']' && top != '[')) {
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

✓ Case 1 ✓ Case 2 ✓ Case 3 ✓ Case 4 ✓ Case 5

Input

Prepare > C++ > Introduction > Variable Sized Arrays Exit Full Screen View

Problem Submissions Leaderboard Discussions Editorial

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.

Constraints

- $1 \le n \le 10^5$
- $1 \le q \le 10^5$
- $1 \le \ k \le 3 \cdot 10^5$
- $n \le \sum k \le 3 \cdot 10^5$
- $0 \le \ i < n$
- $0 \le \ j < k$
- All indices in this challenge are zero-based.
- All the given numbers are non negative and are not greater than 10^6

Output Format

For each pair of i and j values (i.e., for each query), print a single integer that denotes the element located at index j of the array referenced by $a[i]$. There should be a total of q lines of output.

Sample Input

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

Change Theme Language C++11

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

int main() {
    int n, q;
    cin >> n >> q;

    vector<vector<int>> arr(n);


    for(int i = 0; i < n; i++) {
        int k;
        cin >> k;

        arr[i].resize(k);

        for(int j = 0; j < k; j++) {
            cin >> arr[i][j];
        }
    }


    for(int x = 0; x < q; x++) {
        int i, j;
        cin >> i >> j;
        cout << arr[i][j] << endl;
    }

    return 0;
}
```

Line: 10 Col: 5

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

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

Your Output (stdout)

```
5
9
```

Expected Output Download

Prepare > C++ > Classes > Virtual Functions

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

This problem is to get you familiar with virtual functions. Create three classes Person, Professor and Student. The class Person should have data members name and age. The classes Professor and Student should inherit from the class Person.

The class Professor should have two integer members: publications and cur_id. There will be two member functions: getdata and putdata. The function getdata should get the input from the user: the name, age and publications of the professor. The function putdata should print the name, age, publications and the cur_id of the professor.

The class Student should have two data members: marks, which is an array of size 6 and cur_id. It has two member functions: getdata and putdata. The function getdata should get the input from the user: the name, age, and the marks of the student in 6 subjects. The function putdata should print the name, age, sum of the marks and the cur_id of the student.

For each object being created of the Professor or the Student class, sequential id's should be assigned to them starting from 1.

Solve this problem using virtual functions, constructors and static variables. You can create more data members if you want.

Note: Expand the main function to look at how the input is being handled.

Input Format

The first line of input contains the number of objects that are being created. If the first line of input for each object is 1, it means that the object being created is of the Professor class, you will have to input the name, age and publications of the professor.

If the first line of input for each object is 2, it means that the object is of the Student class, you will have to input the name, age and the marks of the student in 6 subjects.

Constraints

$1 \leq len_{name} \leq 100$, where len_{name} is the length of the name.

$1 \leq age \leq 80$

$1 \leq publications \leq 1000$

$0 \leq marks \leq 100$, where marks is the marks of the student in each subject.

Output Format

There are two types of output depending on the object.

If the object is of type Professor, print the space separated name, age, publications and id on a new line.

If the object is of the Student class, print the space separated name, age, the sum of the marks in 6 subjects and id on a new line.

Sample Input

4

Change Theme Language C++11

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
class Person {
public:
    string name;
    int age;
    virtual void getdata() = 0;
    virtual void putdata() = 0;
};

class Professor : public Person {
private:
    int publications;
    static int id_counter;
    int cur_id;
public:
    Professor() {
        cur_id = ++id_counter;
    }
    void getdata() override {
        cin >> name >> age >> publications;
    }
    void putdata() override {
        cout << name << " " << age << " " << publ
    }
};
int Professor::id_counter = 0;

class Student : public Person {
private:
    int marks[6];
```

Line: 54 Col: 29

Upload Code as File

Run Code Submit Code

Test against custom input

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
Walter 56 99
2
Jesse 18 50 48 97 76 34 98
2
Pinkman 22 10 12 0 18 45 50
1
White 58 87
```