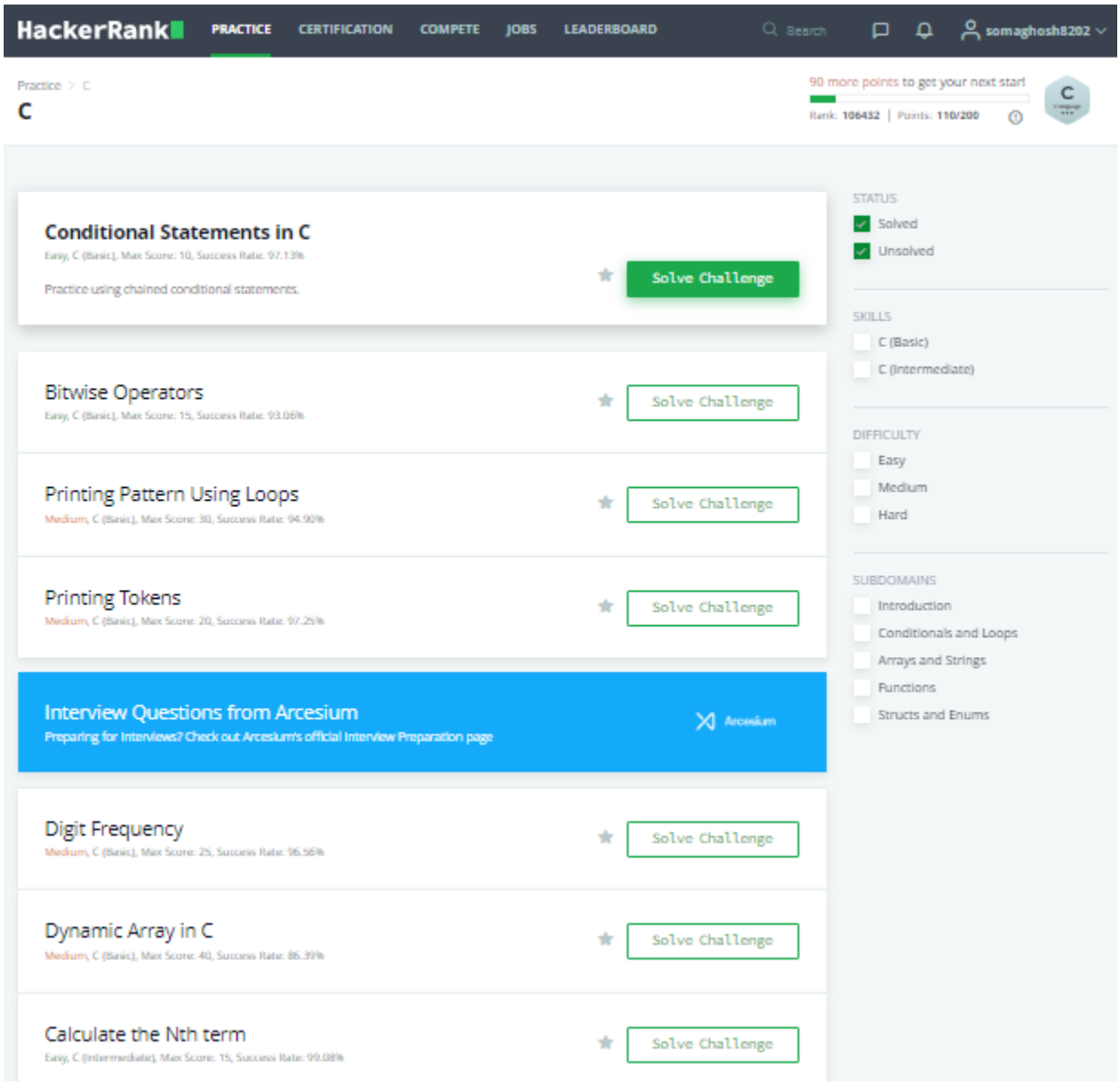
NAME: SOMA **REGISTRATION NUMBER:20BCE2141**



"Hello World!" in C

Problem | Submissions | Leaderboard | Discussions | Editorial

Objective

In this challenge, we will learn some basic concepts of C that will get you started with the language. You will need to use the same syntax to read input and write output in many C challenges. As you work through these problems, review the code stubs to learn about reading from stdin and writing to stdout.

Task

This challenge requires you to print $Hello, World!$ on a single line, and then print the already provided input string to stdout. If you are not familiar with C, you may want to read about the printf() command.

Example

$s = \text{"Life is beautiful"}$

The required output is:

```
Hello, World!
Life is beautiful
```

Function Descriptio

Complete the main() function below.

The main() function has the following input:

- string s: a string

Prints

- *two strings: * "Hello, World!" on one line and the input string on the next line.

Input Format

There is one line of text, s.

Sample Input 0

```
Welcome to C programming.
```

Sample Output 0

```
Hello, World!
Welcome to C programming.
```

Change Theme Language: C

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main()
{

    char s[100];
    printf("Hello, World!\n");
    scanf("%[^\n]%*c", &s);
    printf("%s",s);


    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}

```

Congratulations

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

Next Challenge

Test case 0

Compiler Message

Success

Input (stdin) Download

```
Welcome to C programming.
```

Expected Output Download

```
Hello, World!
Welcome to C programming.
```

Playing With Characters

Objective

This challenge will help you to learn how to take a character, a string and a sentence as input in C.

To take a single character ch as input, you can use `scanf("%c", &ch );` and `printf("%c", ch)` writes a character specified by the argument char to stdout

```
char ch;
scanf("%c", &ch);
printf("%c", ch);
```

This piece of code prints the character ch.

You can take a string as input in C using `scanf("%s", s)`. But, it accepts string only until it finds the first space.

In order to take a line as input, you can use `scanf("%[^\n]%*c", s);` where s is defined as `char s[MAX_LEN]` where MAX_LEN is the maximum size of s. Here, `[]` is the scanset character. `^\n` stands for taking input until a newline isn't encountered. Then, with this `%*c`, it reads the newline character and here, the used $*$ indicates that this newline character is discarded.

Note: The statement: `scanf("%[^\n]%*c", s);` will not work because the last statement will read a newline character, `\n`, from the previous line. This can be handled in a variety of ways. One way is to use `scanf("\n");` before the last statement.

Task

You have to print the character, ch, in the first line. Then print s in next line. In the last line print the sentence, sen.

Input Format

First, take a character, ch as input.

Then take the string, s as input.

Lastly, take the sentence sen as input.

Constraints

Strings for s and sen will have fewer than 100 characters, including the newline.

Output Format

Print three lines of output. The first line prints the character, ch.

The second line prints the string, s.

The third line prints the sentence, sen.

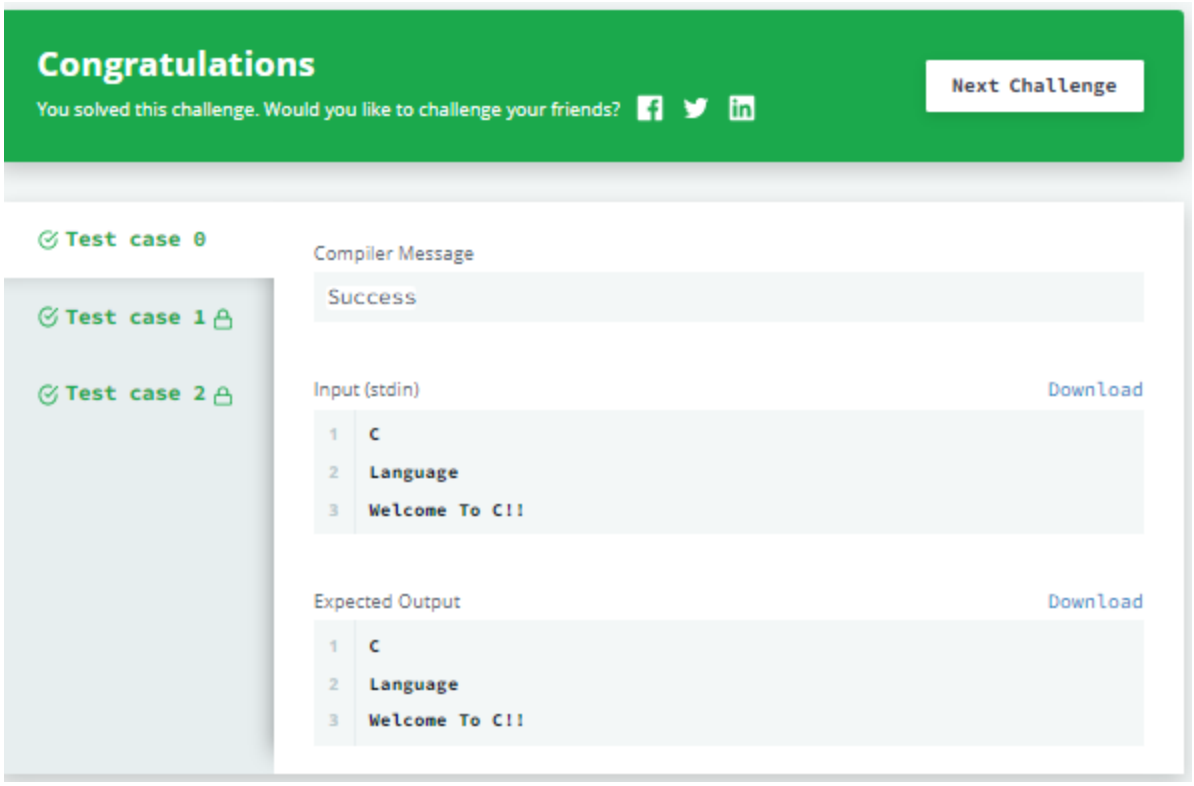
Sample Input 0

```
C
Language
Welcome To C!!
```

Sample Output 0

```
C
Language
Welcome To C!!
```

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main() {

    char ch,s[20],sen[30];
    scanf("%c ", &ch);
    scanf("%s ", &s);
    fgets(sen,30, stdin);
    printf("%c\n",ch);
    printf("%s\n",s);
    printf("%s",sen);

    return 0;
}

```


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Compiler Message
Success
Input (stdin)
Download
1 C
2 Language
3 Welcome To C!!
Expected Output
Download
1 C
2 Language
3 Welcome To C!!

Sum and Difference of Two Numbers

Problem | Submissions | Leaderboard | Discussions | Editorial

Objective

The fundamental data types in c are int, float and char. Today, we're discussing int and float data types.

The `printf()` function prints the given statement to the console. The syntax is `printf("format string",argument_list);`. In the function, if we are using an integer, character, string or float as argument, then in the format string we have to write `%d` (integer), `%c` (character), `%s` (string), `%f` (float) respectively.

The `scanf()` function reads the input data from the console. The syntax is `scanf("format string",argument_list);`. For ex: The `scanf("%d",&number)` statement reads integer number from the console and stores the given value in variable $number$.

To input two integers separated by a space on a single line, the command is `scanf("%d %d", &n, &m)`, where n and m are the two integers.

Task

Your task is to take two numbers of int data type, two numbers of float data type as input and output their sum:

1. Declare 4 variables: two of type int and two of type float.
2. Read 2 lines of input from stdin (according to the sequence given in the 'Input Format' section below) and initialize your 4 variables.
3. Use the $+$ and $-$ operator to perform the following operations:
   - Print the sum and difference of two int variable on a new line.
   - Print the sum and difference of two float variable rounded to one decimal place on a new line.

Input Format

The first line contains two integers.
The second line contains two floating point numbers.

Constraints

- $1 \leq$ integer variables $\leq 10^4$
- $1 \leq$ float variables $\leq 10^4$

Output Format

Print the sum and difference of both integers separated by a space on the first line, and the sum and difference of both float (scaled to 1 decimal place) separated by a space on the second line.

Sample Input

```
10 4
4.0 2.0
```

Sample Output

```
14 6
6.0 2.0
```

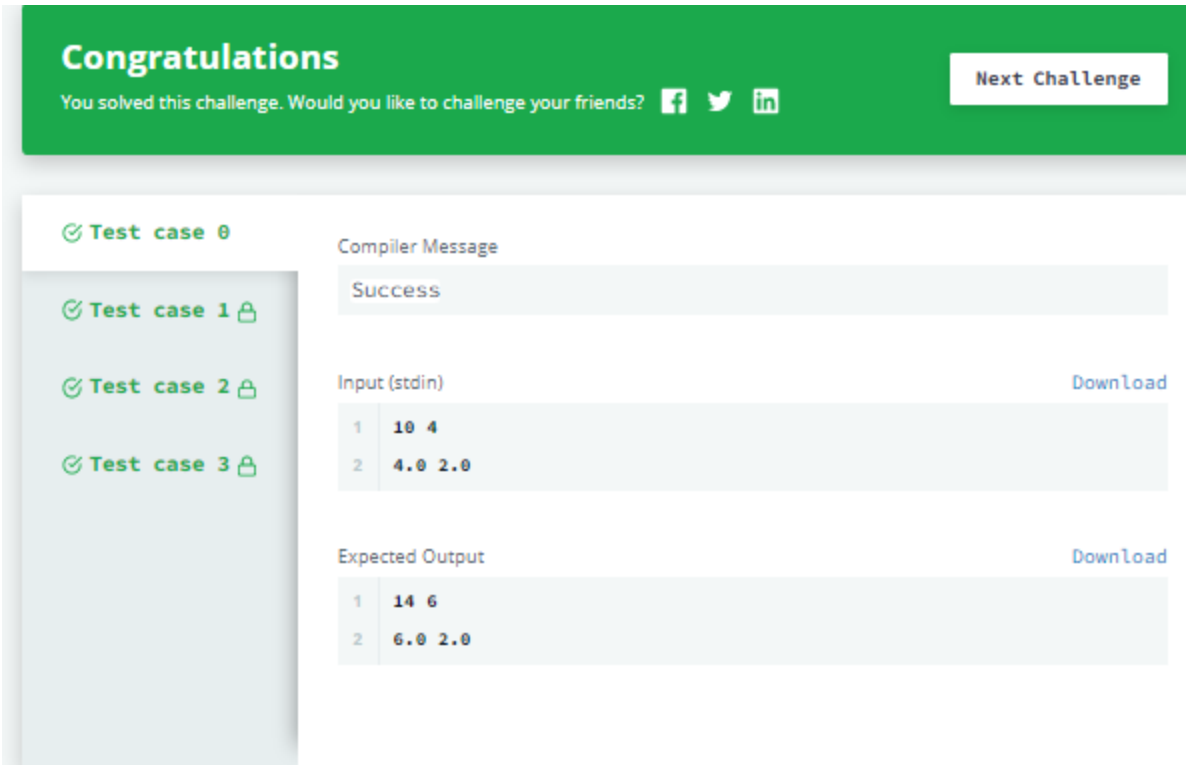
Explanation

When we sum the integers 10 and 4, we get the integer 14. When we subtract the second number 4 from the first number 10, we get 6 as their difference.

When we sum the floating-point numbers 4.0 and 2.0, we get 6.0. When we subtract the second number 2.0 from the first number 4.0, we get 2.0 as their difference.

Change Theme Language: C

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main()
{

    int num1,num2,sum=0,s1=0;
    float num3,num4,sum1=0.0,s2=0.0;
    scanf("%d %d",&num1,&num2);
    scanf("%f %f",&num3,&num4);
    sum=num1+num2;
    sum1=num3+num4;
    s1=num1-num2;
    s2=num3-num4;
    printf("%d %d\n",sum,s1);
    printf("%0.1f %0.1f",sum1,s2);
    return 0;
}

```


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Compiler Message
Success
Input (stdin)
Download
1 10 4
2 4.0 2.0
Expected Output
Download
1 14 6
2 6.0 2.0

Functions in C

Problem | Submissions | Leaderboard | Discussions | Editorial

Objective

In this challenge, you will learn simple usage of functions in C. Functions are a bunch of statements grouped 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.

A sample 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 read four variables and return the sum of them 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;
}
```

```
+= : Add and assignment operator. It adds the right operand to the left operand and assigns
the result to the left operand.

a += b is equivalent to a = a + b;
```

Task

Write a function `int max_of_four(int a, int b, int c, int d)` which reads four arguments and returns the greatest of them.

Note

There is not built in `max` function in C. Code that will be reused is often put in a separate function, e.g. `int max(x, y)` that returns the greater of the two values.

Input Format

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

Output Format

Print the greatest of the four integers.

Note: I/O will be automatically handled.

Sample Input

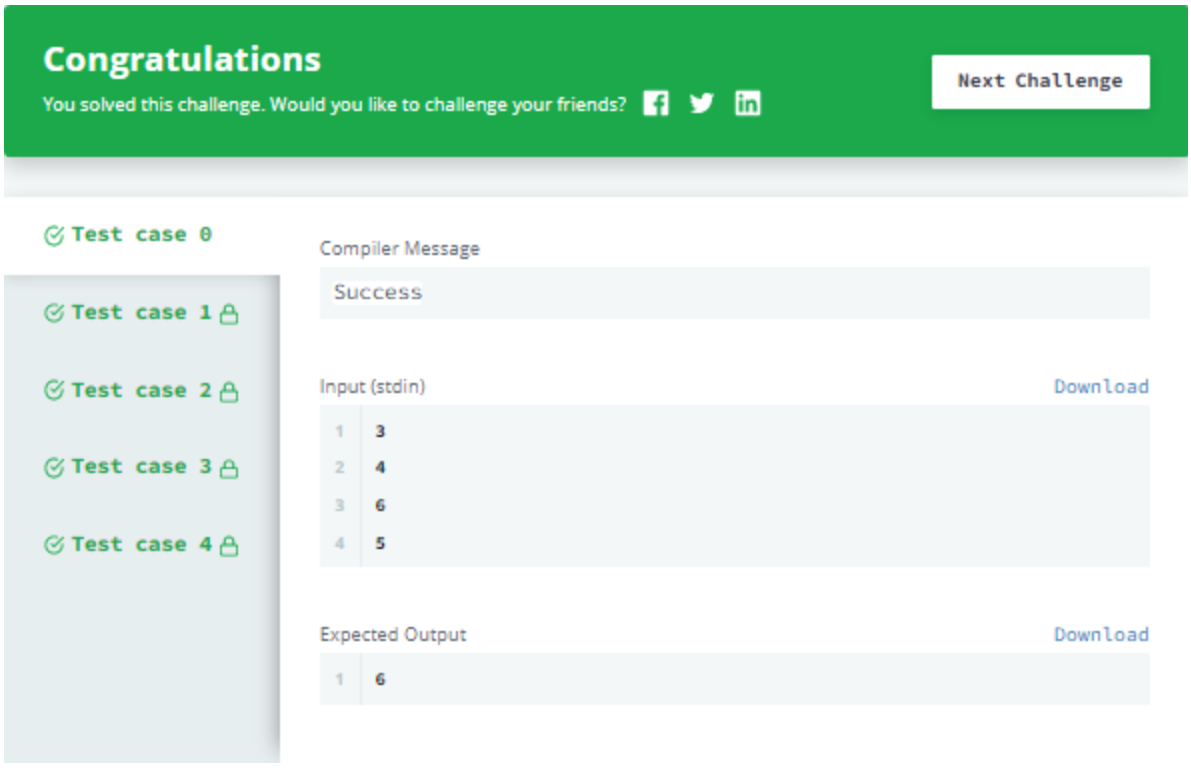
```
3
4
6
5
```

Sample Output

```
6
```

Change Theme C

```c
#include <stdio.h>
/*
Add `int max_of_four(int a, int b, int c, int d)` here.
*/
int max_of_four(int x, int y, int z, int w);
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;
}
int max_of_four(int x, int y, int z, int w)
{
    int arr[4],temp=0;
    arr[0]=x;arr[1]=y;arr[2]=z;arr[3]=w;
    for(int i=0;i<4;i++)
    {
        if(arr[i]>temp)
        {
            temp=arr[i];
        }
    }
    return temp;
}
```


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Compiler Message
Success
Input (stdin)
Download
1 3
2 4
3 6
4 5
Expected Output
Download
1 6

Pointers in C

Problem | Submissions | Leaderboard | Discussions | Editorial

Objective

In this challenge, you will learn to implement the basic functionalities of pointers in C. A pointer in C is a way to share a memory address among different contexts (primarily functions). They are primarily used whenever a function needs to modify the content of a variable that it does not own.

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 various functions. For example, $int* p = \&val$ will assign the memory address of val to pointer p. To access the content of the memory to which the pointer points, prepend it with a `*`. For example, `*p` will return the value reflected by val and any modification to it will be reflected at the source (val).

```
    void increment(int *v) {
    (*v)++;
}
    int main() {
    int a;
    scanf("%d", &a);
    increment(&a);
    printf("%d", a);
    return 0;
}
```

Task

Complete the function `void update(int *a,int *b)`. It receives two integer pointers, int* a and int* b. Set the value of a to their sum, and b to their absolute difference. There is no return value, and no return statement is needed.

- $a' = a + b$
- $b' = |a - b|$

Input Format

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

Output Format

Modify the two values in place and the code stub main() will print their values.

Note: Input/ouput will be automatically handled. You only have to complete the function described in the 'task' section.

Sample Input

```
4
5
```

Sample Output

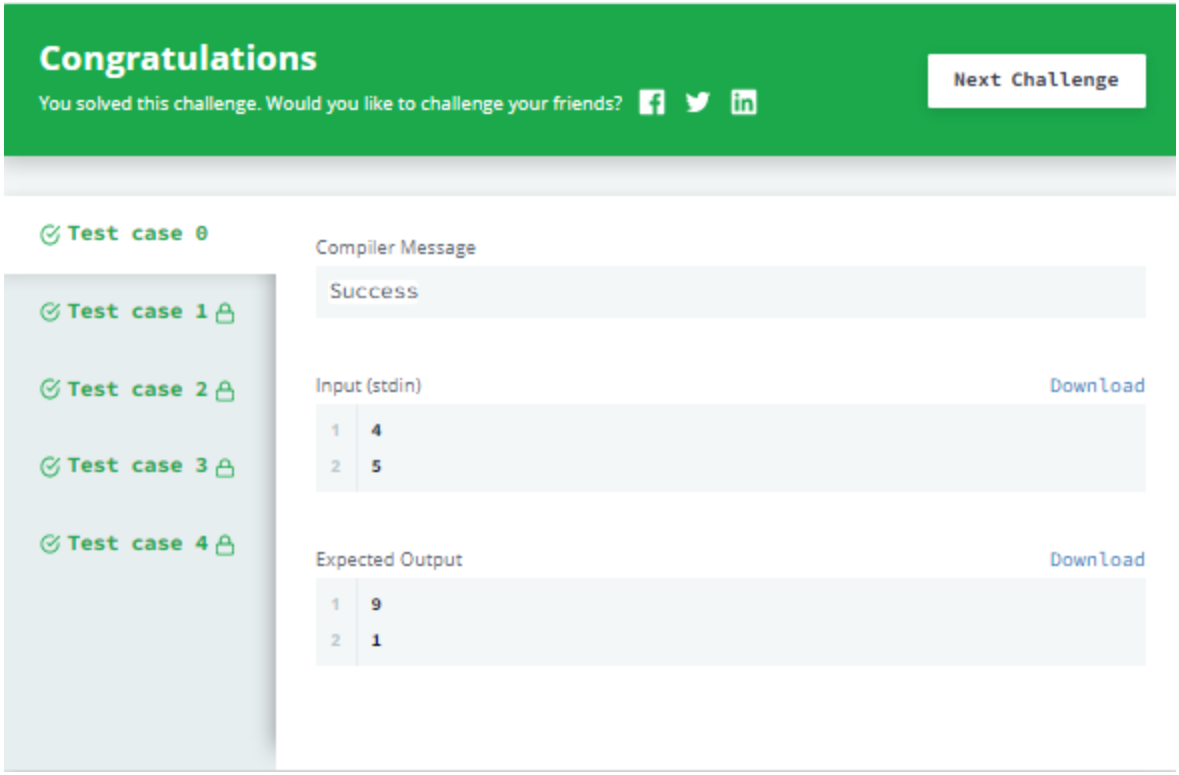
```
9
1
```

Explanation

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

Change Theme Language: C

```c
#include <stdio.h>
#include<stdlib.h>

void update(int *a,int *b)
{
    // Complete this function
    int sum,diff;
    sum=(*a+*b);
    diff= abs(*a-*b);
    *a=sum;
    *b=diff;
}

int main() {
    int a, b;
    int *pa = &a, *pb = &b;

    scanf("%d %d", &a, &b);
    update(pa, pb);
    printf("%d\n%d", a, b);
    return 0;
}
```


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Compiler Message
Success
Input (stdin)
Download
1 4
2 5
Expected Output
Download
1 9
2 1

For Loop in C

Problem | Submissions | Leaderboard | Discussions | Editorial

Objective

In this challenge, you will learn the usage of the for loop, which is a programming language statement which allows code to be executed until a terminal condition is met. They can even repeat forever if the terminal condition is never met.

The syntax for the `for` loop 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.

The following loop initializes i to 0, tests that i is less than 10, and increments i at every iteration. It will execute 10 times.

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

Task

For each integer n in the interval $[a, b]$ (given as input) :

- 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".

Input Format

The first line contains an integer, a.
The seond line contains an integer, b.

Constraints

$1 \leq a \leq b \leq 10^6$

Constraints

$1 \leq a \leq b \leq 10^6$

Output Format

Print the appropriate English representation,`even`, or `odd`, based on the conditions described in the 'task' section.

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

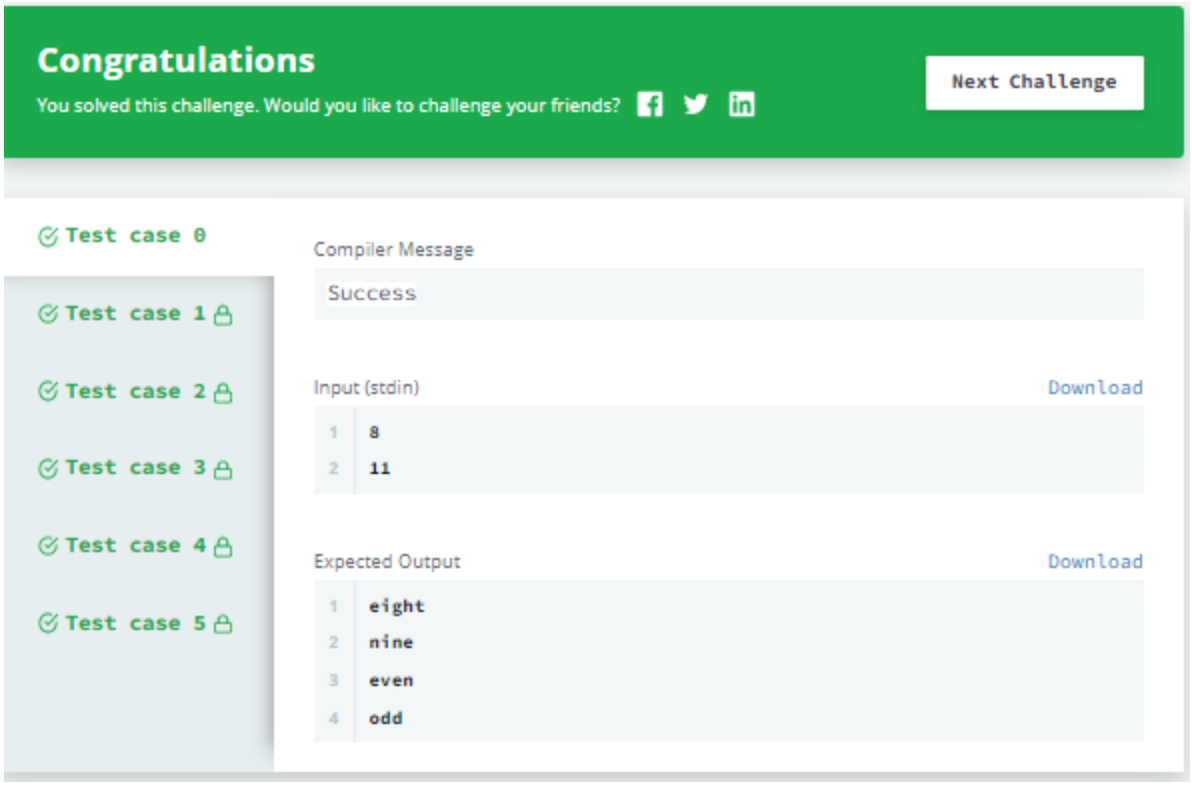
Sample Input

```
8
11
```

Sample Output

```
eight
nine
even
odd
```

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>



int main()
{
    int a, b;
    scanf("%d\n%d", &a, &b);
    // Complete the code.
    for(int i=a;i<=b;i++)
    {
        if(i>=1 && i<=9)
        {
            if(i==1)
            printf("one\n");
            else if(i==2)
            printf("two\n");
            else if(i==3)
            printf("three\n");
            else if(i==4)
            printf("four\n");
            else if(i==5)
            printf("five\n");
            else if(i==6)
            printf("six\n");
            else if(i==7)
            printf("seven\n");
            else if(i==8)
            printf("eight\n");
            else if(i==9)
            printf("nine\n");
        }
        else if(i>9)
        {
            if(i%2==0)
            printf("even\n");
            else if(i%2!=0)

            else if(i%2!=0)
            printf("odd\n");
        }
    }
    return 0;
}
```


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
8
11
Expected Output
Download
eight
nine
even
odd

Sum of Digits of a Five Digit Number

Problem | Submissions | Leaderboard | Discussions | Editorial

Objective

The modulo operator, `%`, returns the remainder of a division. For example, `4 % 3 = 1` and `12 % 10 = 2`. The ordinary division operator, `/`, returns a truncated integer value when performed on integers. For example, `5 / 3 = 1`. To get the last digit of a number in base 10, use 10 as the modulo divisor.

Task

Given a five digit integer, print the sum of its digits.

Input Format

The input contains a single five digit number, n.

Constraints

$10000 \leq n \leq 99999$

Output Format

Print the sum of the digits of the five digit number.

Sample Input 0

```
10564
```

Sample Output 0

```
16
```

Change Theme Language: C

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main() {

    int n,sum=0,d1,d2;
    scanf("%d", &n);
    while(n!=0)
    {
        d1=n%10;
        sum=sum+d1;
        n=n/10;
    }
    printf("%d",sum);
    //Complete the code to calculate the sum of the five digits on n.

    return 0;
}
```

Congratulations

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

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Input (stdin) Download

```
10564
```

Expected Output Download

```
16
```

1D Arrays in C

Problem | Submissions | Leaderboard | Discussions | Editorial

An array is a container object that holds a fixed number of values of a single type. To create an array in C, we can do `int arr[n];`. Here, arr, is a variable array which holds up to 10 integers. The above array is a static array that has memory allocated at compile time. A dynamic array can be created in C, using the malloc function and the memory is allocated on the heap at runtime. To create an integer array, arr of size n, `int *arr = (int*)malloc(n * sizeof(int))`, where arr points to the base address of the array. When you have finished with the array, use `free(arr)` to deallocate the memory.

In this challenge, create an array of size n dynamically, and read the values from stdin. Iterate the array calculating the sum of all elements. Print the sum and free the memory where the array is stored.

While it is true that you can sum the elements as they are read, without first storing them to an array, but you will not get the experience working with an array. Efficiency will be required later.

Input Format

The first line contains an integer, n.
The next line contains n space-separated integers.

Constraints

$1 \leq n \leq 1000$
$1 \leq a[i] \leq 1000$

Output Format

Print the sum of the integers in the array.

Sample Input 0

```
6
16 13 7 2 1 12
```

Sample Output 0

```
51
```

Sample Input 1

```
7
```

Sample Input 0

```
6
16 13 7 2 1 12
```

Sample Output 0

```
51
```

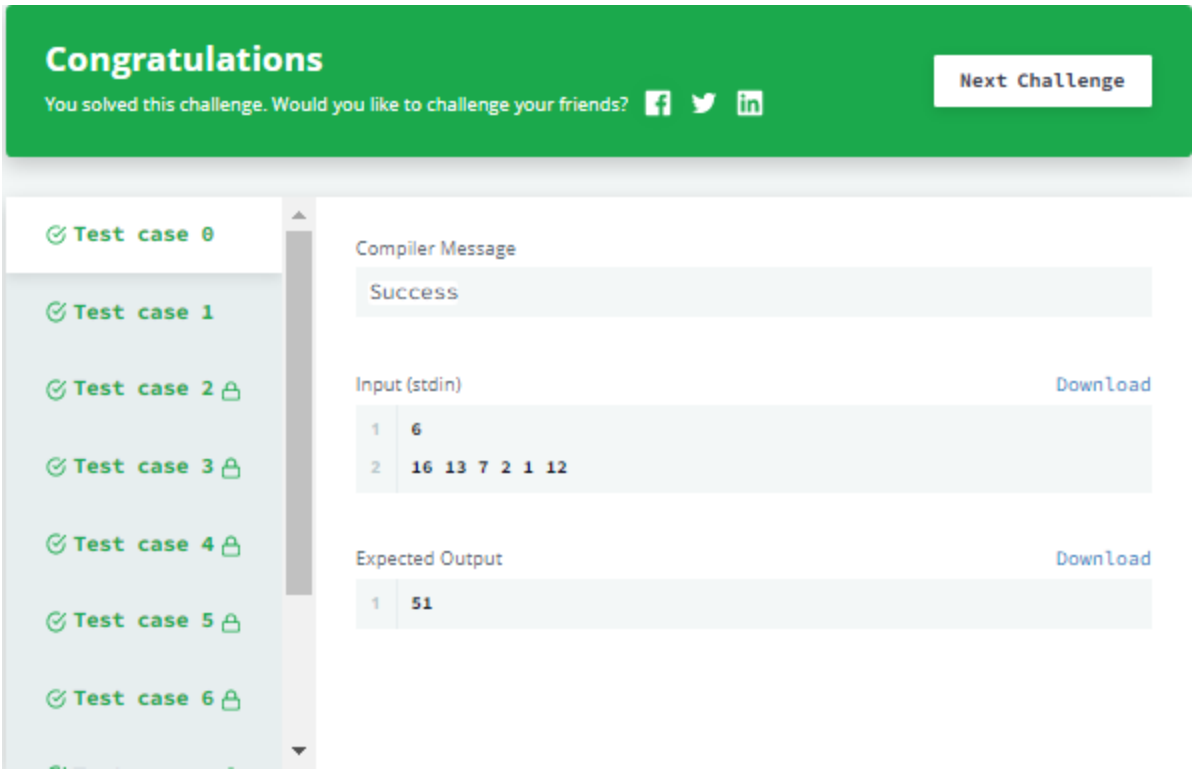
Sample Input 1

```
7
1 13 15 20 12 13 2
```

Sample Output 1

```
76
```

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main()
{
    int sum=0;
    int l;
    scanf("%d\n",&l);
    int *arr=(int*)malloc(l* sizeof(int));
    for(int i=0;i<l;i++)
    {
        scanf("%d",arr+i);
    }
    for(int j=0;j<l;j++)
    {
        sum=sum+(*(arr+j));
    }
    printf("%d",sum);


    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}

```


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1 6
2 16 13 7 2 1 12
Expected Output
Download
1 51

Array Reversal

Problem | Submissions | Leaderboard | Discussions | Editorial

Given an array, of size n, reverse it.

Example: If array, $arr = [1, 2, 3, 4, 5]$, after reversing it, the array should be, $arr = [5, 4, 3, 2, 1]$.

Input Format

The first line contains an integer, n, denoting the size of the array. The next line contains n space-separated integers denoting the elements of the array.

Constraints

$1 \leq n \leq 1000$

$1 \leq arr_i \leq 1000$, where arr_i is the i^{th} element of the array.

Output Format

The output is handled by the code given in the editor, which would print the array.

Sample Input 0

```
6
16 13 7 2 1 12
```

Sample Output 0

```
12 1 2 7 13 16
```

Explanation 0

Given array, $arr = [16, 13, 7, 2, 1, 12]$. After reversing the array, $arr = [12, 1, 2, 7, 13, 16]$

Sample Input 1

```
7
1 13 15 20 12 13 2
```

Sample Output 0

```
12 1 2 7 13 16
```

Explanation 0

Given array, arr = $[16, 13, 7, 2, 1, 12]$. After reversing the array, arr = $[12, 1, 2, 7, 13, 16]$

Sample Input 1

```
7
1 13 15 20 12 13 2
```

Sample Output 1

```
2 13 12 20 15 13 1
```

Sample Input 2

```
8
15 5 16 15 17 11 5 11
```

Sample Output 2

```
11 5 11 17 15 16 5 15
```

Change Theme Language: C

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int num, *arr, i;
    scanf("%d", &num);
    arr = (int*) malloc(num * sizeof(int));
    for(i = 0; i < num; i++)
    {
        scanf("%d", arr + i);
    }


    /* Write the logic to reverse the array. */

    for(i = num-1; i>=0; i--)
        printf("%d ", *(arr + i));
    return 0;
}

```

Congratulations

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

Next Challenge

- Test case 0
- Test case 1
- Test case 2
- Test case 3
- Test case 4
- Test case 5
- Test case 6

Compiler Message

Success

Input (stdin) Download

```
6
16 13 7 2 1 12
```

Expected Output Download

```
12 1 2 7 13 16
```

Students Marks Sum

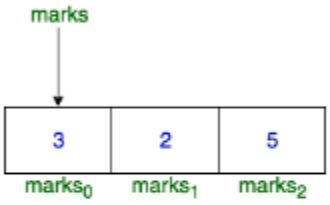
Problem | Submissions | Leaderboard | Discussions | Editorial

You are given an array of integers, $marks$, denoting the marks scored by students in a class.

- The alternating elements $marks_0$, $marks_2$, $marks_4$ and so on denote the marks of boys.
- Similarly, $marks_1$, $marks_3$, $marks_5$ and so on denote the marks of girls.

The array name, $marks$, works as a pointer which stores the base address of that array. In other words, $marks$ contains the address where $marks_0$ is stored in the memory.

For example, let $marks = [3, 2, 5]$ and $marks$ stores 0x7fff9575c05f. Then, 0x7fff9575c05f is the memory address of $marks_0$.



Complete the function, `marks_summation(int* marks, char gender, int number_of_students)` which returns the total sum of:

- marks of boys if $gender = b$
- marks of girls if $gender = g$

The locked code stub reads the elements of $marks$ along with $gender$. Then, it calls the function `marks_summation(marks, gender, number_of_students)` to get the sum of alternate elements as explained above and then prints the sum.

Input Format

- The first line contains $number\_of\_students$, denoting the number of students in the class, hence the number of elements in $marks$.
- Each of the $number\_of\_students$ subsequent lines contains $marks_i$.
- The next line contains $gender$.

Constraints

- $1 \leq number\_of\_students \leq 10^3$
- $1 \leq marks_i \leq 10^3$ (where $0 \leq i < number\_of\_students$)
- $gender$ = g or b

Output Format

The output should contain the sum of all the aternate elements in $marks$ as explained above.

Sample Input 0

```
3
3
2
5
b
```

Sample Output 0

```
8
```

Explanation 0

$marks$ = [3, 2, 5] and $gender$ = b.

So, $marks_0 + marks_2 = 3 + 5 = 8$.

Sample Input 1

```
5
1
2
3
4
5
g
```

Sample Output 1

```
6
```

Explanation 1

$marks$ = [1, 2, 3, 4, 5] and $gender$ = g

So, $sum = marks_1 + marks_3$ = $2 + 5$ = 8.

Sample Input 2

```
1
5
g
```

Sample Output 2

```
0
```

Explanation 2

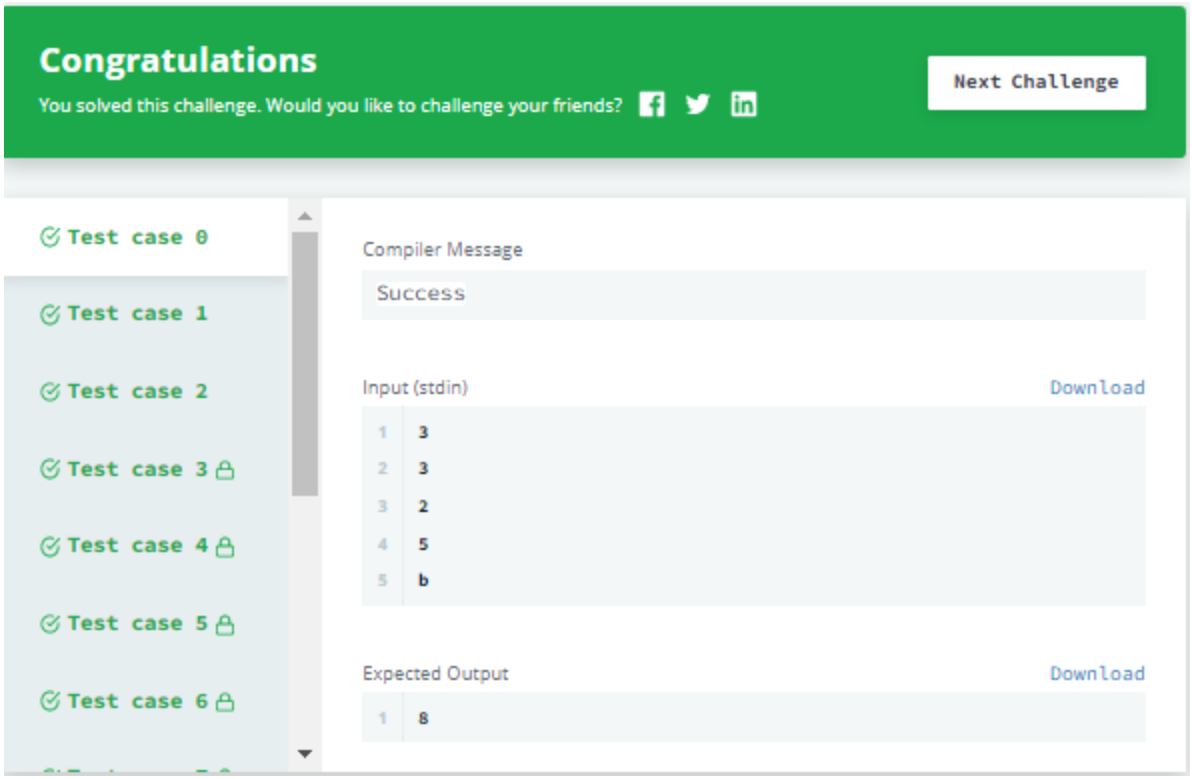
$marks$ = [5] and $gender$ = g

Here, $marks_1$ does not exist. So, sum = 0.

Change Theme Language: C

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

//Complete the following function.

int marks_summation(int* marks, int number_of_students, char gender) {
  //Write your code here.
  int sum=0;
  for (int i = (gender == 'b' ? 0 : 1); i < number_of_students; i = i + 2)
        sum += *(marks + i);

    return sum;



}

int main() {
    int number_of_students;
    char gender;
    int sum;

    scanf("%d", &number_of_students);
    int *marks = (int *) malloc(number_of_students * sizeof (int));

    for (int student = 0; student < number_of_students; student++) {
        scanf("%d", (marks + student));
    }

    scanf(" %c", &gender);
    sum = marks_summation(marks, number_of_students, gender);
    printf("%d", sum);
    free(marks);

    return 0;
}
```

Line: 5 Col: 1


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
3
3
2
5
b
Expected Output
Download
8