WEEK 1-01

Overview of C, Constants, Variables and Data Types

Roll no : 240701095

Name : Darshini.H

Attempt 1

Status	Finished
Started	Monday, 23 December 2024, 5:33 PM
Completed	Wednesday, 9 October 2024, 2:52 PM
Duration	75 days 2 hours

PROGRAM 1: This is a simple challenge to help you practice printing to stdout.

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.

Input Format

You do not need to read any input in this challenge.

Output Format

Print ***Hello, World!*** to stdout.

Sample Output 1

```
Hello, World
```

CODE:

Answer: (penalty regime: 0 %)

```
1  #include<stdio.h>
2  int main()
3  {
4   printf("Hello, World!");
5   return 0;
6  }
```

OUTPUT:

	Expected	Got	
✓	Hello, World!	Hello, World!	✓

Passed all tests! ✓

PROGRAM 2: 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 chars [**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: After inputting the character and the string, inputting the sentence by the above mentioned statement won't work. This is because, at the end of each line, a new line character(\n) is present. So, the statement: scanf("%[^\n] %*c", s); will not work because the last statement will read a newline character from the previous line. This can be handled in a variety of ways and one of them being: 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

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 1
C
program
Programming using
C **Sample Output**
1
C
program Programming using C

CODE:

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
char ch;
 scanf("%c",&ch);
 printf("%c", ch);
}
```

OUTPUT:

	Input	Expected	Got	
✓	c	c	c	✓

Passed all tests! ✓

PROGRAM 3: 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

CODE:

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int a,b;
    float c,d;
    scanf("%d %d" ,&a,&b);
    scanf("%f %f" ,&c,&d);
    printf("%d %d\n",a+b,a-b);
    printf("%.1f %.1f\n",c+d,c-d);
    return 0;
}
```

OUTPUT:

	Input	Expected	Got	
✓	10 4 4.0 2.0	14 6 6.0 2.0	14 6 6.0 2.0	✓
✓	20 8 8.0 4.0	28 12 12.0 4.0	28 12 12.0 4.0	✓

Passed all tests! ✓