Programming Fundamental

Lab Manual - Week 05

Introduction

Welcome to your favorite programming Lab. In this lab manual, we shall work together to learn and implement new programming concepts.

Skills to be learned:

- Solve problems by using User-defined Functions and Pre-defined Functions.
- Understanding and Implementing Void and Value Returning Functions.

Let's do some coding.

Skill: Solve problems by using User-defined Functions and Pre-defined Functions.

Introduction

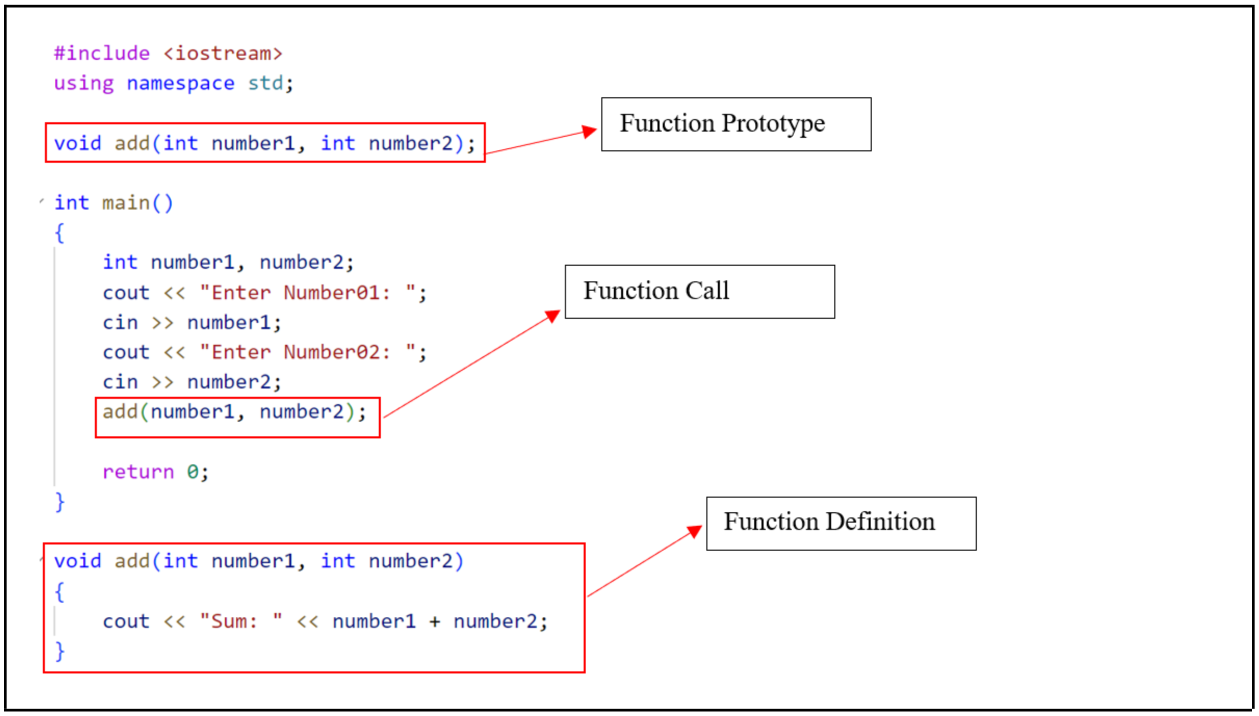
By this week, you have learned how to write a program that contains functions. The functions are the reusable pieces of code that can be executed multiple times using the function call. However, there are two types of functions that include

- User-defined functions
- Pre-defined functions

User-defined functions

The type of functions created and implemented by the user are referred to as user-defined functions. Consider the following example for understanding the user-defined functions.

Task 01(WP): Create a function that takes two numbers from the user and prints their sum on the screen.



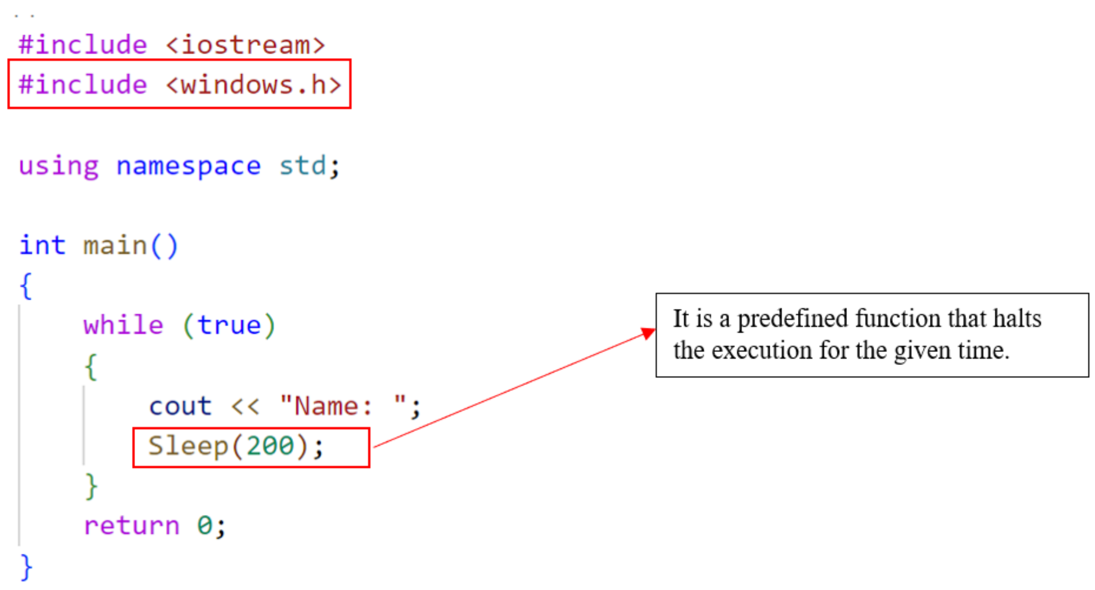
```
#include <iostream>
using namespace std;

void add(int number1, int number2);

int main()
{
    int number1, number2;
    cout << "Enter Number01: ";
    cin >> number1;
    cout << "Enter Number02: ";
    cin >> number2;
    add(number1, number2);

    return 0;
}

void add(int number1, int number2)
{
    cout << "Sum: " << number1 + number2;
}
```

All such functions that are defined and created by the user are referred to as user defined functions.

Pre-defined functions

These functions are pre-defined by the Programmers of the language and are not implemented by the user himself. In most cases, the user just includes a file that provides that Pre-Defined function.
Consider the example below for further understanding:

Task 02(WP): Write a program that halts the execution for 200 milliseconds using the Sleep function.



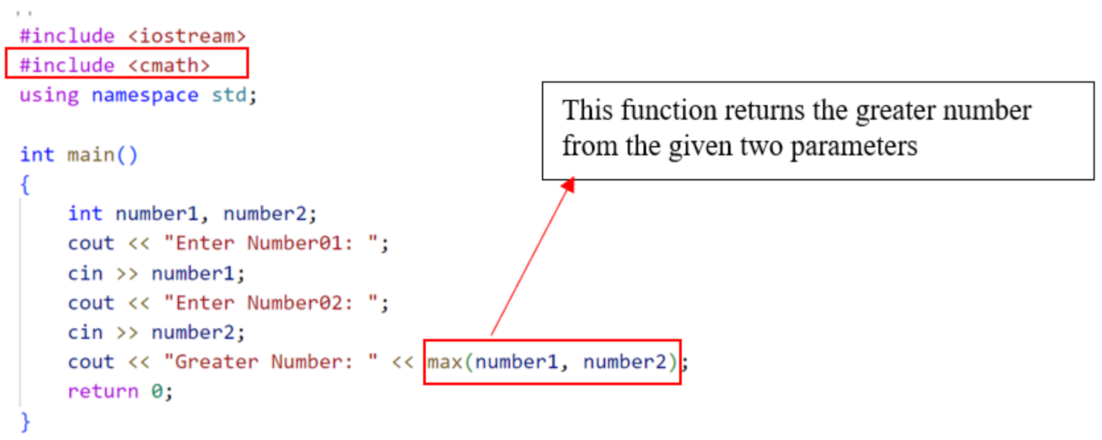
```cpp
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    while (true)
    {
        cout << "Name: ";
        Sleep(200);
    }
    return 0;
}
```

It is a predefined function that halts the execution for the given time.

Notice that we have included **windows.h** file in our program and we can use the **Sleep()** function **that accepts parameters** as time in milliseconds and halts the execution of the program for that much time.
Sleep() is a pre-defined function that we have used in this example.

There is a similar library of pre-defined mathematical functions that we can use in our programs to solve different mathematical problems. Consider the below-mentioned tasks for better understanding.

Task 03(CL): Write a c++ program that takes two numbers from the user and prints the greater number on the screen.



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

int main()
{
    int number1, number2;
    cout << "Enter Number01: ";
    cin >> number1;
    cout << "Enter Number02: ";
    cin >> number2;
    cout << "Greater Number: " << max(number1, number2);
    return 0;
}
```

This function returns the greater number from the given two parameters

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

Notice that we have included the library **cmath** in our program that enables us to use the **max()** function that accepts two numbers as parameters and returns the greater. By now, you should have the idea that the **max()** is a predefined function which has been defined in the **cmath** library.
Congratulations !!!! You have learned the difference between user-defined and pre-defined functions.

Task 04(OP): Write a program that takes two numbers as input from the user and prints the minimum out of two on the screen. Hint: **min**(number1,number2)
Task 05(OP): Write a c++ program that takes two numbers from the user and takes the power of the first number as the second number entered by the user. Hint: **pow**(number1,number2)
Test Case:

Number1: 5 **Number2:** 3	Power: 125
Number1: 2 **Number2:** 6	Power: 64

Task 06(OP): Write a c++ program that takes a number from the user as input and print its square root on the screen. Hint: **sqrt**(number)

Task 07(CP): Write a C++ Program to Find the sin of a Number using a pre-defined sin() function.

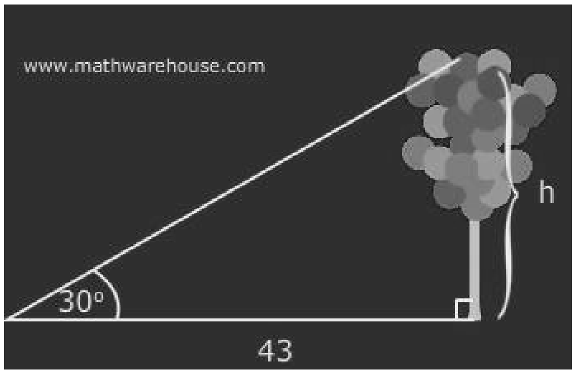
Following are a few other pre-defined in the cmath library. Try out the for yourself.

Function	**Description**
cbrt(x)	Returns the cube root of x
ceil(x)	Returns the value of x rounded up to its nearest integer
floor(x)	Returns the value of x rounded down to its nearest integer

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

cos(x)	Returns the cosine of x
sin(x)	Returns the sine of x (x is in radians)
tan(x)	Returns the tangent of an angle

Task 08(CP):
The angle of elevation from a point 43 feet from the base of a tree on level ground to the top of the tree is 30° (30 degree). Write a C++ program to calculate the height of the tree?
Hint:
Use pre-defined sin(), cos() and tan() functions which are defined in cmath library.
Remember: 1 radian = 57.2958 degrees
Output:
Correct Answer is : h = 24.8261



Task 09(CP):
Write a C++ program that calculates the value of x for the following equation
5x2 + 6x + 1 = 0
Hint:
Just put the values of a, b and c into the Quadratic Formula, and do the calculations.
Output:
Answer: x = −0.2 or x = −1

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Skill: Solve problems by using User-defined Functions and Pre-defined Functions

Conclusion

Functions	Description
User-defined Functions	These are the type of the functions that are defined and implemented by the user. The user has to do the following in order to use user-defined functions 1. Define **Function Prototype** 2. Define **Function Definition** 3. Make **Function Call** when the functionality is required.
Pre-defined Functions	These are the type of functions that are already defined and created by programmers. The user has to do the following for using these functions. 1. **Include** the library that has the definition of the function. 2. Make a **Function Call** when the functionality is required.

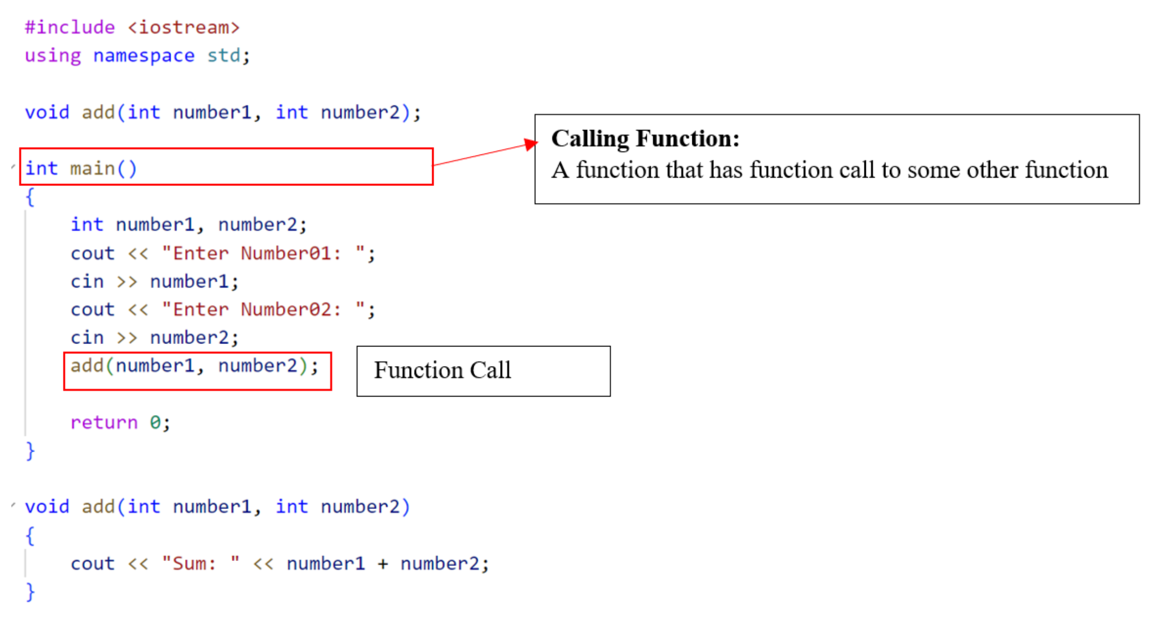
Skill: Solve problems by using User-defined Functions and Pre-defined Functions

Skill: Understanding and Implementing Void and Value Returning Functions

Introduction

Functions **may return a value** to the **calling function** depending on the **user requirement**.

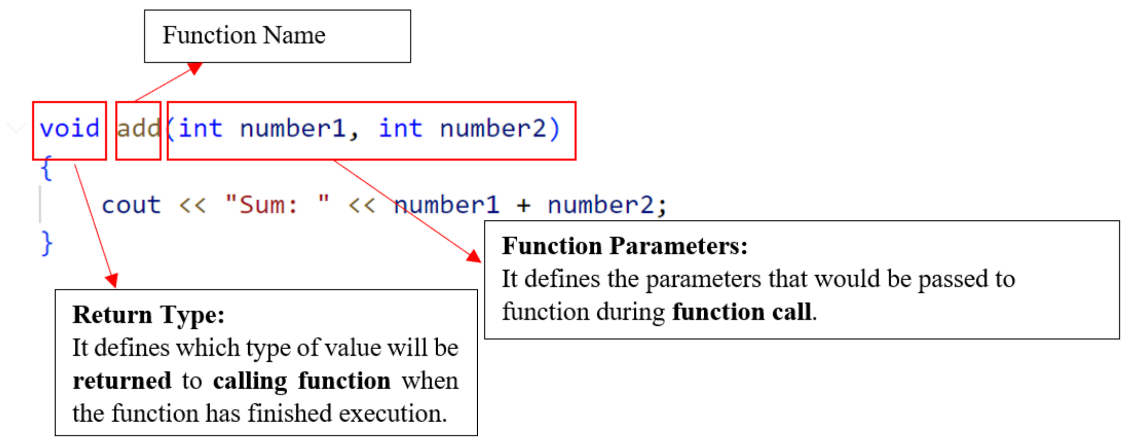
Consider the previously used example mentioned below



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

void add(int number1, int number2);

int main()
{
    int number1, number2;
    cout << "Enter Number01: ";
    cin >> number1;
    cout << "Enter Number02: ";
    cin >> number2;
    add(number1, number2);

    return 0;
}

void add(int number1, int number2)
{
    cout << "Sum: " << number1 + number2;
}
```

In this example, the **calling function** is **main()** that has an **add() function call**.

So, how do we know, if a function is returning any value or not? Recall this concept again



```
void add(int number1, int number2)
{
    cout << "Sum: " << number1 + number2;
}
```

Function Name

Function Parameters:
It defines the parameters that would be passed to function during **function call**.

Return Type:
It defines which type of value will be **returned** to **calling function** when the function has finished execution.

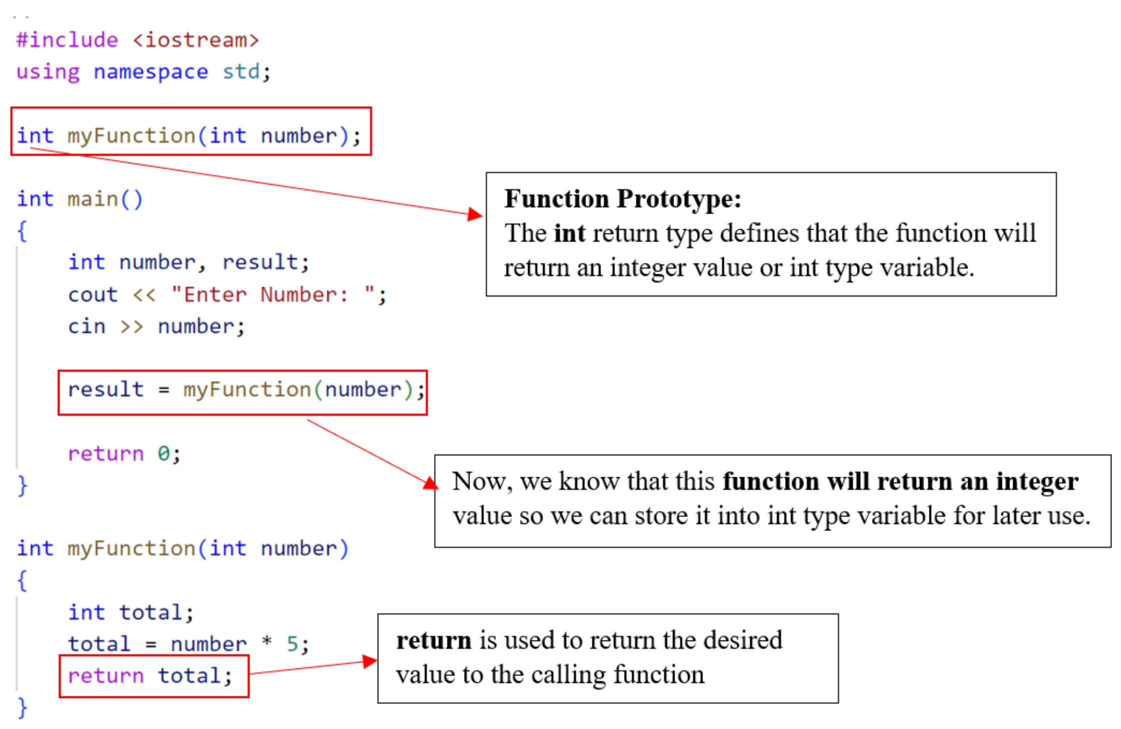
Function Return Type defines the type of value that will be returned to the calling function when the function has finished execution.

Following are the type of values that can be returned by a function

Datatype	**Description**
void	This datatype means that the **function will not return** any value
bool	This datatype means that the function will return **a boolean type value or variable**. True/False
int	This means that the function will return **an integer type value or variable**.
float	This means that the function will return a **float type value or variable**.
char	This means that the function will return a **character type value or variable**.
string	This means that the function will return a **string type value or variable**.

Task 01(WP): Write a function that takes a number from the user and returns it after multiplying it with 5.

Skill: Understanding and Implementing Void and Value Returning Functions



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

int myFunction(int number);

int main()
{
    int number, result;
    cout << "Enter Number: ";
    cin >> number;

    result = myFunction(number);

    return 0;
}

int myFunction(int number)
{
    int total;
    total = number * 5;
    return total;
}
```

Function Prototype:
The **int** return type defines that the function will return an integer value or int type variable.

Now, we know that this **function will return an integer** value so we can store it into int type variable for later use.

return is used to return the desired value to the calling function

1. The **int** return type in the function prototype defines that the function will return an integer variable.
2. Notice, that the function is returning an integer type variable **total**
3. Now, in the calling function, the **returned value** is stored in the **result** so it may be used in the future.

Great Work Students ‼ You have understood the various return types in the functions. Add this one to your skill set. 👏

Conclusion

Functions have a return type that defines **which kind of data will be sent back to the calling function** once the function has completed execution.

Task 02(CP):

Write a function for checking whether the alphabet entered by the user is in small case or in capital case (Suppose user can only enter ‘A’ or ‘a’). Make a function that takes 1 Character as input, does processing according to the input and then returns the string.

Skill: Understanding and Implementing Void and Value Returning Functions

String is “You have entered Capital A” if the user enters ‘A’, otherwise “You have entered small A”.

Task 03(CP): Create a function that takes a number as an argument and returns true or false depending on whether the number is symmetrical or not. A number is symmetrical when it is the same as its reverse. (**The user can enter three digit number only**)
Test Cases:

```
IsSymmetrical(12567) ➞ false
IsSymmetrical(12321) ➞ True
```

Task 04(CP): Create a function that determines whether a number is Oddish or Evenish. A number is Oddish if the sum of all of its digits is odd, and a number is Evenish if the sum of all of its digits is even. If a number is Oddish, return "Oddish". Otherwise, return "Evenish". (**The user can enter five digit number only**)

```
OddishOrEvenish(43) ➞ "Oddish"
// 4 + 3 = 7
// 7 % 2 = 1

OddishOrEvenish(373) ➞ "Oddish"
// 3 + 7 + 3 = 13
// 13 % 2 = 1

OddishOrEvenish(4433) ➞ "Evenish"
// 4 + 4 + 3 + 3 = 14
// 14 % 2 = 0
```

Task 05(CP): Write a program that inputs hours and minutes of a 24-hour day and calculates what will be the time after 15 minutes. Print the result in hh:mm format. Hours are always between 0 and 23, and minutes are always between 0 and 59. Hours are written with one or two digits.

Input	Output
1 46	2:1
0	0:16

Skill: Understanding and Implementing Void and Value Returning Functions

01	
23 59	0:14
11 08	11:23
12 49	13:4

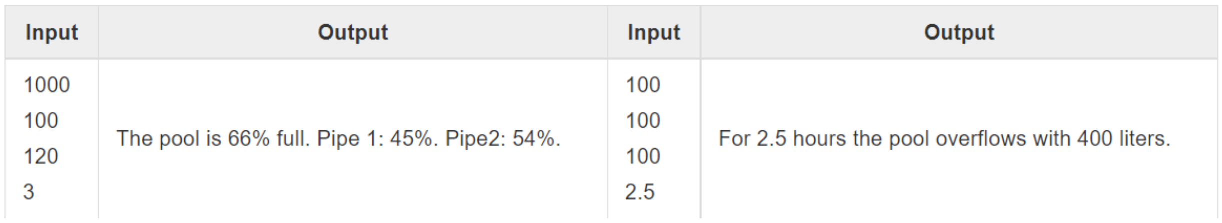
Task 06(CP): Write a program that converts a number in the range of [1 ... 99] into text (in English). (**The user can enter two-digit numbers only except 11-19**)

25	twenty five
42	forty two
6	six

Task 07(CP):
A pool with volume V fills up via two pipes. Each pipe has a certain flow rate (the liters of water, flowing through a pipe for an hour). A worker starts the pipes simultaneously and goes out for N hours. Write a program that finds the state of the pool the moment the worker comes back.

Input data: V – the volume of the pool in liters - an integer in the range of [1 … 10000]. P1 – the flow rate of the first pipe per hour – an integer in the range of [1 … 5000]. P2 – the flow rate of the second pipe per hour – an integer in the range of [1 … 5000]. H – the hours that the worker is absent – a floating-point number in the range of [1.0 … 24.00].
Output data: ● To what extent the pool has filled up and how much percent has each pipe contributed. All percent values must be formatted to an integer (without rounding). ○ "The pool is [x]% full. Pipe 1: [y]%. Pipe 2: [z]%." ● If the pool has overflown – with how many liters it has overflown for the given time – a floating-point number. ○ "For [x] hours the pool overflows with [y] liters."

Skill: Understanding and Implementing Void and Value Returning Functions



Input	Output	Input	Output
1000 100 120 3	The pool is 66% full. Pipe 1: 45%. Pipe2: 54%.	100 100 100 2.5	For 2.5 hours the pool overflows with 400 liters.

Good Luck and Best Wishes !!

Happy Coding ahead :)