```
// Author -  Krishna Kumari


/* Functios - is a block of code or sub-program which performs a specific task
    multiple times when we call it.


    # ISSUES When functions are not used
    1] Lengthy and bulky code
    2] Buggy
    3] Zero readability
    4] Zero reuseability


    Syntax --->
    return_type function_name()
    {
        //func body
    }
    int main()
       {
           //func body
    }

    These curly braces defines the scope
    of the function.

    # Declaration: the return type, the name of the function, and parameters (if any)

       # Definition: the body of the function (code to be executed)

       # Calling: to invoke the function we must have to call it.


       NOTE: We must define func before calling it.If we to define
       after calling it then we must declare it above main function
       otherwise it will give error.
*/



#include<iostream>
using namespace std;

void print(string name)
{
       for(int i = 0; i < 2; i++)
       {
              cout << name << endl;
       }
}
int main()
{
       print("Harsh"); // calling
       print("Dixit");
       return 0;
}

//-----------------------------------------
```

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

void MyFunction(int num)  // declaration
{
        for (int i = 0; i < num; i++)   // definition
        {
                cout << "I am inside function... " << endl;
        }
}
int main()
{
        MyFunction(5);  // calling
        return 0;
}

//------------------------------------------------


/* There are 2 types of functions

    1] Which return some value ---> int , char , string , bool , array

    2] which returns nothing ---> void
*/

// Function which return int

#include<iostream>
using namespace std;

int findMaximum(int num1,int num2 , int num3)
{

        int maximum = max(max(num1,num2),num3);
        return maximum;
}

int main()
{
    int ans = findMaximum(10,23,3);
    cout << "Maximum b/w 3 numbers is: " << ans;

        return 0;
}


// Function which returns nothing

#include<iostream>
using namespace std;

void printSum(int num1,int num2)
{
        cout << (num1 + num2) << endl;
}

int main()
{
    printSum(10,23);
```

```
       return 0;
}

//------------------------------------------------------

/* FUNCTION CALL STACK ->

   1] Tracks function calls
   2] Local variable -> check upon input variable
   3] Tracks which func is called by which another func
   4] Return value


    Stack --> works on (Last-in first-out)LIFO principle.
              same as plate that are stack on each other in
              home or marriages.


In functions call stack --->

    1] The first entry in the function call stack must be for main function.

    2] Whenever we get a function call an entry should be added for that in
       in function call stack.

    3] Whenever function body or scope ends that entry should be removed from
       function call stack.

*/

#include<iostream>
using namespace std;

void Function1()
{
       cout << "I am inside Function1. " << endl;
}

void Function2()
{
       cout << "I am inside Function2. " << endl;
}

void Function3()
{
       cout << "I am inside Function3. " << endl;
}

int main()
{
       Function1();
       Function3();
       Function2();
}
```