NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCE

CL 1004 - Object Oriented Programming Lab

Semester: Spring 2022

Lab 02

Instructor: Muhammad Sudais

Email: muhammad.sudais.v@nu.edu.pk

Outline:

- ✓ C++ Data Types
- ✓ Functions
  - Predefined Functions
  - User-Defined Functions
  - Function Parameters
    - Value Parameters
    - Reference Parameters
- ✓ Structures

C++ DATA TYPES:

Variables are containers for storing data values.

In C++, there are different types of variables (defined with different keywords), for example:

- int - stores integers (whole numbers), without decimals, such as 123 or -123
- double - stores floating point numbers, with decimals, such as 19.99 or -19.99
- char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- string - stores text, such as "Hello World". String values are surrounded by double quotes
- bool - stores values with two states: true or false

Examples:

```
int myNum = 5;               // Integer (whole number)
float myFloatNum = 5.99;     // Floating point number
double myDoubleNum = 9.98;   // Floating point number
char myLetter = 'D';         // Character
bool myBoolean = true;       // Boolean
string myText = "Hello";     // String
```

C++ FUNCTIONS:

A variable in C++ must be a specified data type.

A function is a block of code which only runs when it is called.

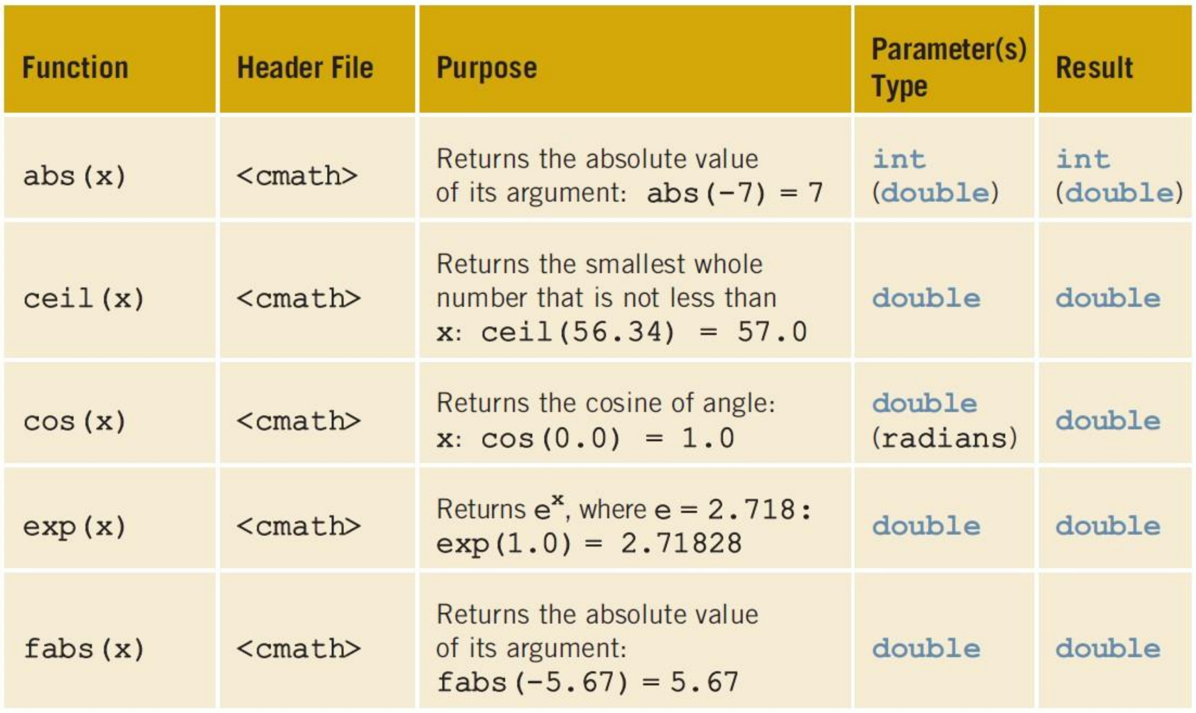
You can pass data, known as parameters, into a function.

Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

Predefined Functions

Functions in C++ are similar to that of in Algebra. For Example, every function has a name and depending on the value specified by the user. It does some computation and gives an output (if any).

Some examples for the built in functions are given below.



Function	Header File	Purpose	Parameter(s) Type	Result
`abs(x)`	`<cmath>`	Returns the absolute value of its argument: `abs(-7) = 7`	`int` (`double`)	`int` (`double`)
`ceil(x)`	`<cmath>`	Returns the smallest whole number that is not less than `x: ceil(56.34) = 57.0`	`double`	`double`
`cos(x)`	`<cmath>`	Returns the cosine of angle: `x: cos(0.0) = 1.0`	`double` (`radians`)	`double`
`exp(x)`	`<cmath>`	Returns e^x, where e = `2.718`: `exp(1.0) = 2.71828`	`double`	`double`
`fabs(x)`	`<cmath>`	Returns the absolute value of its argument: `fabs(-5.67) = 5.67`	`double`	`double`

Example Code for Predefined Functions:

```
//How to use predefined functions.
#include <iostream>
#include <cmath>
#include <cctype>
using namespace std;
int main ()
{
        int x;
        double u, v;
        u = 4.2; //Line 1
        v = 3.0; //Line 2
        cout << "Line 3: " << u << " to the power of " << v << " = " << pow (u, v) << endl; //Line 3
        cout << "Line 4: 5.0 to the power of 4 = " << pow (5.0, 4) << endl; //Line 4
        u = u + pow (3.0, 3); //Line 5
        cout << "Line 6: u = " << u << endl; //Line 6 x = -15; //Line 7
        cout << "Line 8: Absolute value of " << x << " = " <<abs(x) << endl; //Line 8
        return 0;
}
```

Sample Run:
Line 3: 4.2 to the power of 3 = 74.088
Line 4: 5.0 to the power of 4 = 625
Line 6: u = 31.2
Line 8: Absolute value of -15 = 15

User Defined Functions

User defined functions are classified into two categories.

- **Value-Returning Functions:** These functions return a value of a specific type using *return* statement.
- **Void Functions:** These functions *do not* use a *return* statement to return a value.

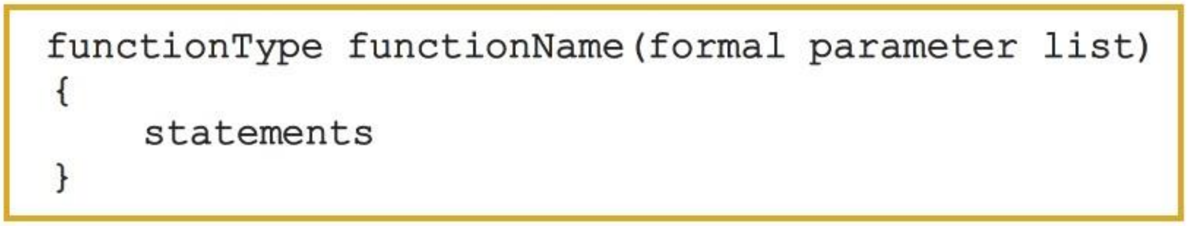
Value-Returning Functions:

Value-returning functions can be utilized in one of three ways:

- Save the value for further calculation.
- Use the value in some calculation.
- Print the value.

These function can be called in following scenarios.

- In an assignment statement.
- As a parameter in a function call.
- In an output statement. Syntax:

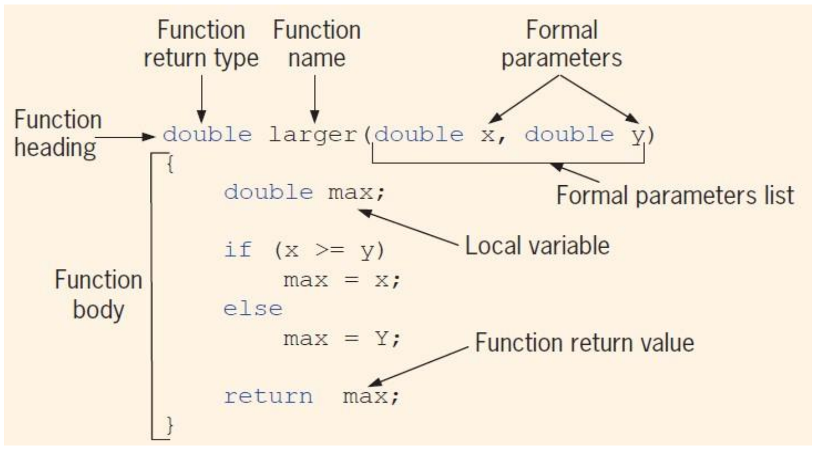


```
functionType functionName(formal parameter list)
{
    statements
}
```

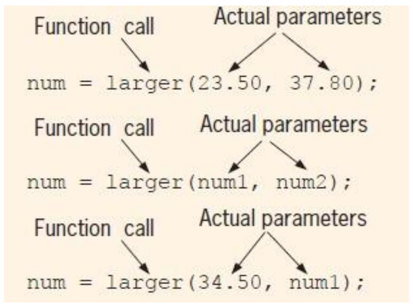
Example Code for Value Returning Functions:

```
double larger (double x, double y)
{
double max;
if (x >= y)
        max = x;
else
        max = y;
return max;
}
```

Components:



```
double larger(double x, double y)
{
    double max;

    if (x >= y)
        max = x;
    else
        max = Y;

    return  max;
}
```



```
num = larger(23.50, 37.80);
num = larger(num1, num2);
num = larger(34.50, num1);
```

<u>Usage in cout:</u>

```
cout << "The larger of 5 and 6 is " << larger (5, 6) << endl; //Line 1
cout << "The larger of " << one << " and " << two << " is " << larger (one, two) << endl; //Line 2
cout << "The larger of " << one << " and 29 is " << larger (one, 29) << endl; //Line 3
maxNum = larger (38.45, 56.78); //Line 4
```

<u>Usage in another Function:</u>

```
double compareThree (double x, double y, double z)
{
        return larger (x, larger (y, z));
}
```

<u>Some Peculiarities in Functions:</u>

A function with a returning must return a value. Consider the following function.

```
int secret (int x)
{
        if (x > 5) //Line 1
                return 2 * x; //Line 2
}
```

Now in this function when the condition of *x > 5* is not met then the return statement will not execute. Hence resulting in no return value. In this case function may or may not return a meaningful value. It may result some strange value.

Function Parameters:

When passing parameters to a function there are ways to do that.

1. Value Parameters (Pass by Value).
2. Reference Parameters (Pass by Reference).

Value Parameters:

When passing value parameters in a function, the parameter is copied into the corresponding formal parameter. There is no connection between the actual and formal parameter values, this means that these parameters cannot be used to pass the result back to the calling function.

Example Code for Value Parameters:

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

void funcValueParam (int num);

int main ()
{
int number = 6; //Line 1
cout << "Line 2: Before calling the function " << "funcValueParam, number = " << number << endl; //Line 2
funcValueParam(number); //Line 3
cout << "Line 4: After calling the function " << "funcValueParam, number = " << number << endl; //Line 4
return 0;
}

void funcValueParam (int num)
{
    cout << "Line 5: In the function funcValueParam, " << "before changing, num = " << num <<endl; //Line 5
    num = 15; //Line 6
    Value Parameters | 367
    cout << "Line 7: In the function funcValueParam, " << "after changing, num = " << num << endl; //Line 7
}
```

Sample Run:

```
Line 2: Before calling the function funcValueParam, number = 6
Line 5: In the function funcValueParam, before changing, num = 6
Line 7: In the function funcValueParam, after changing, num = 15
Line 4: After calling the function funcValueParam, number = 6
```

Reference Parameters:

When a reference parameter is passed in a function, it receives the address (memory location) of the actual parameter. Reference parameters can change the value of the actual parameter.

Reference parameters are useful in following situations.

1- When the value of the actual parameter needs to be changed.
2- When you want to return more than one value from a function.
3- When passing the address would save memory space and time relative to copying a large amount of data.

Example code for Reference Parameters:

```
//This program reads a course score and prints the
//associated course grade. #include <iostream> using
namespace std;

void getScore (int& score); void printGrade (int score); int
main ()
{
        int courseScore;
        cout << "Line 1: Based on the course score, \n" << " this program computes the " << "course grade."
        << endl; //Line 1
        getScore(courseScore); //Line 2
        printGrade(courseScore); //Line 3
        return 0;
}
void getScore (int& score)
{
        cout << "Line 4: Enter course score: "; //Line 4
        cin >> score; //Line 5
        cout << endl << "Line 6: Course score is " << score << endl; //Line 6
}
void printGrade (int cScore)
{
        cout << "Line 7: Your grade for the course is "; //Line 7
        if (cScore >= 90) //Line 8
                cout << "A." << endl;
        else if (cScore >= 80)
                cout << "B." << endl;
        else if (cScore >= 70)
                cout << "C." << endl;
        else if (cScore >= 60)
                cout << "D." << endl;
        else
                cout << "F." << endl;
}
```

Sample Run: In this sample run, the user input is shaded. Line 1: Based on the course score, this program computes the course grade.

```
Line 4: Enter course score: 85
Line 6: Course score is 85
Line 7: Your grade for the course is B.
```

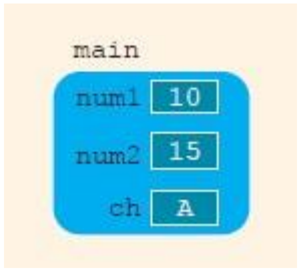
Example code for Value & Reference Parameters:

```
//Example 7-6: Reference and value parameters
#include <iostream> using namespace std;

void funOne (int a, int& b, char v);
void funTwo (int& x, int y, char& w);

int main ()
{
        int num1, num2;
        char ch;
        num1 = 10; //Line 1
        num2 = 15; //Line 2
        ch = 'A'; //Line 3
        cout << "Line 4: Inside main: num1 = " << num1 << ", num2 = " << num2 << ", and ch = " << ch <<
        endl; //Line 4

        funOne (num1, num2, ch); //Line 5
        cout << "Line 6: After funOne: num1 = " << num1 << ", num2 = " << num2 << ", and ch = " << ch <<
        endl; //Line 6

        funTwo (num2, 25, ch); //Line 7
        cout << "Line 8: After funTwo: num1 = " << num1 << ", num2 = " << num2 << ", and ch = " << ch <<
        endl; //Line 8
        return 0;
}

void funOne (int a, int& b, char v)
{
        int one;
        one = a; //Line 9
        a++; //Line 10
        b = b * 2; //Line 11
        v = 'B'; //Line 12
        cout << "Line 13: Inside funOne: a = " << a << ", b = << b << ", v = " << v << ", and one = " << one <<
        endl; //Line 13
}

void funTwo (int& x, int y, char& w)
{
        x++; //Line 14
        y = y * 2; //Line 15
        w = 'G'; //Line 16
        cout << "Line 17: Inside funTwo: x = " << x << ", y = " << y << ", and w = " << w << endl; //Line 17
}
```

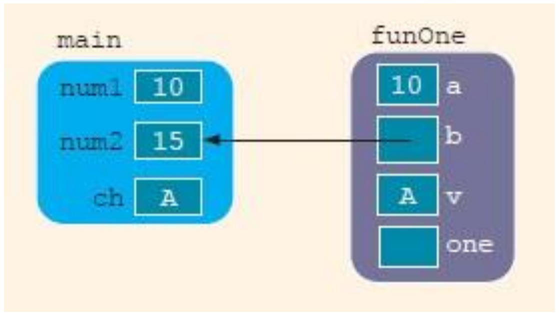
Sample Run:
Line 4: Inside main: num1 = 10, num2 = 15, and ch = A
Line 13: Inside funOne: a = 11, b = 30, v = B, and one = 10
Line 6: After funOne: num1 = 10, num2 = 30, and ch = A
Line 17: Inside funTwo: x = 31, y = 50, and w = G
Line 8: After funTwo: num1 = 10, num2 = 31, and ch = G

REPRESENTATION OF VARIABLES:

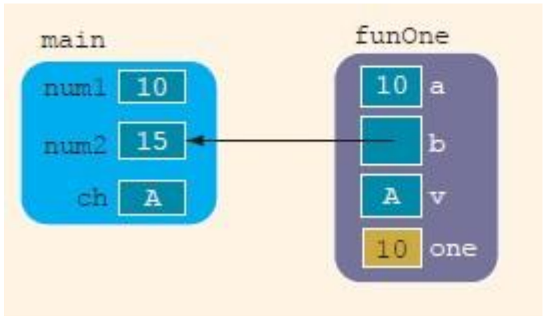
After Line 3:



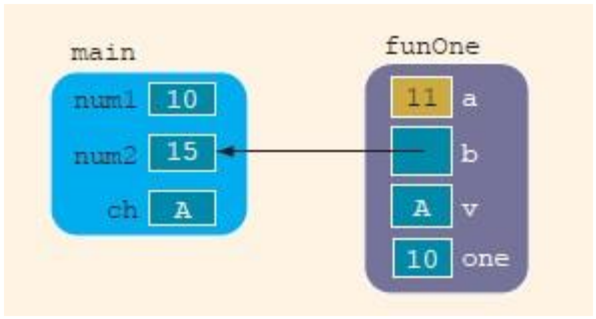
Before Line 9: (In Function *funOne*)



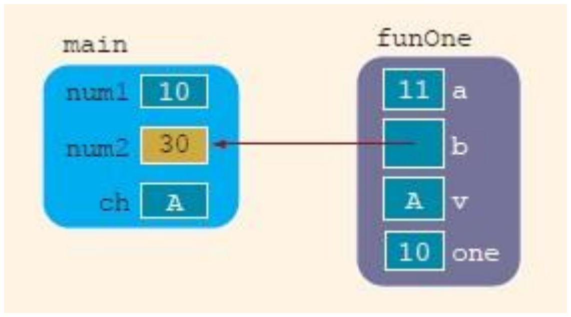
After Line 9:



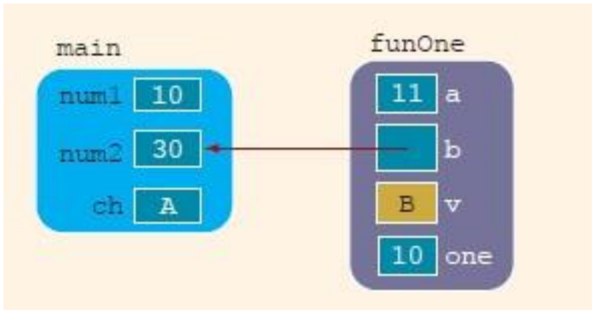
After Line 10:



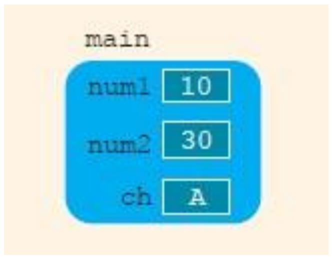
After Line 11:



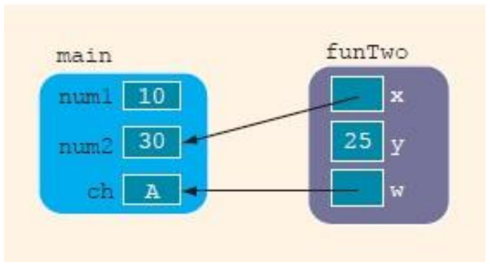
After Line 12:



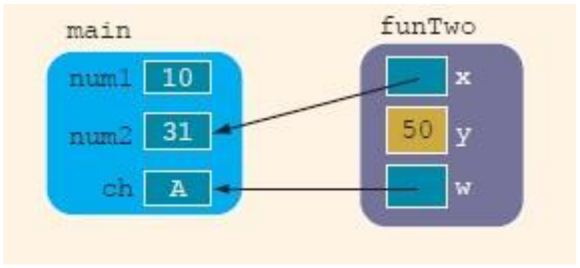
On Line 6: (When the function ends)



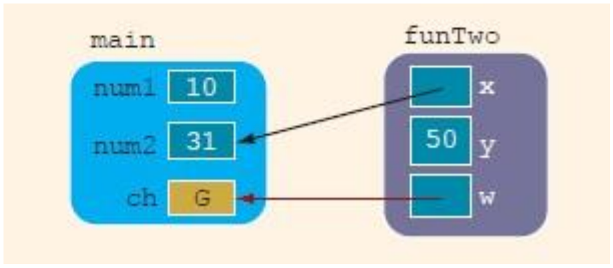
After Line 14: (In Function *funTwo*)



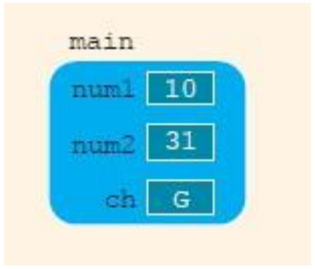
After Line 15:



After Line 16:



At Line 8 (When Function Ends):



STRUCTURES:

- A structure is a collection of fixed number of components in which the components are accessed by name. The components may be of different types.
- Components of a structure are called members of the structure.
- To declare a structure, you need to use the "**struct**" keyword along with the structure name.
    - The **struct** block then contains all the members, which are variables of different/same type.

```
struct structName
{
     dataType1 varName1;
     dataType2 varName2;
                 . . .
     dataTypen varNamen; }
```

- To declare an object of the structure you need to use the **name of structure** and then the name of **structure object.**

```
struct studentStruct
{
string firstName;
string lastName;
char courseGrade;
int testScore;
double GPA;
};
studentStruct Obj;
```

Example Code for Structure:

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

struct studentType
{
string firstName;
string lastName;
char courseGrade;
int courseScore;
double GPA;
};

 int main() {
  studentType newStudent;
  cout << "Enter Details for the Student";
  cout << "Enter Student's First Name : ";
  cin >> newStudent.firstName;
  cout << "Enter Student's Last Name : ";
  cin >> newStudent.lastName;
  cout << "Enter Student's Course Grade : ";
  cin >> newStudent.courseGrade;
  cout << "Enter Student's Course Score : ";
  cin >> newStudent.courseScore;
  cout << "Enter Student's GPA : ";
  cin >> newStudent.GPA;
  cout << newStudent.firstName << endl;
  cout << newStudent.lastName << endl;
  cout << newStudent.courseGrade << endl;
  cout << newStudent.courseScore << endl;
  cout << newStudent.GPA << endl;
}
```

Sample Run: In this sample run, the user input is shaded.

```
Enter Details for the Student
Enter Student's First Name : First_Name
Enter Student's Last Name : Last_Name
Enter Student's Course Grade : A
Enter Student's Course Score : 84
Enter Student's GPA : 2.0
```

First_Name
Last_Name
A
84
2

Exercise

Task - 01:

Write a function which takes a character as input and returns its ASCII equivalent.

Hint: int ConvertToASCII(char x)

Input: A

Output: 65

Task - 02:

Given the formula below:

$$\text{Area} = \pi r^2$$
$$\text{Circumference} = 2\pi r$$
$$\text{Diameter} = 2r$$
$$\text{Pie} = \pi r^2 \times \text{degree}/360$$

Write the following functions which take radius as input. You have to take input in main function and pass it as the arguments of below functions.

1. Area
2. Circumference
3. Diameter
4. Area of Pie: float AreaofPie (int radius, int degreeOfPie)

Task - 03:

Define a struct, **pet**, with two components: **petName** of type **string, petType** of type **string** and **adopt** of type **bool**. Write a program to help a local petstore to automate its adoption system. The program should do the following:

A. Show the customer the different types of pets.
B. Ask the user to select 3 pets from adoptable pets and add them in his pet cot (an array of names or the structure)
C. Show the list of adopted pets.

Assume that the petStore offers the following pets:

Name	Type	Adopt
Kitty	Cat	True
Krypto	Dog	True
Koku	Bird	False
Blu	Fish	True
Felly	Cat	False
Cane	Dog	False
Fluffy	Hamster	True

Task - 04:

Write a program that declares a struct to store the data of a baseball player (player's name, number of home runs, and number of hits). Declare an array of 5 components to store the data of 5 baseball players. Your program must be able to search the array to find the index of a specific player using a search function.

Hint:

Search(list, playerName);

Task - 05:

Rewrite, Complete and Run the codes below. Resolve the errors (if any).

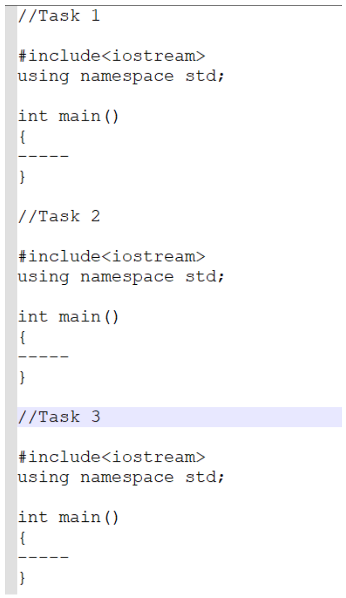
```
#include<iostream>
using namespace std;
void swap(int &x, int____)
{
Write the function logic
}

int main()
{
int firstNumber,secondNumber;
cin >> firstNumber;
cin >> secondNumber;

  cout << "Before swap:  " << "\n";
  cout << firstNumber << " "<< secondNumber << "\n";

 Write Swap FunctionCall Here

  cout << "After swap: " << "\n";
  cout << firstNumber << " "<< secondNumber << "\n";

  return 0;
}
```

```
#include<iostream>
using namespace std;
int main()
{
int students;
int marks;
cout<< "Enter Students Count";
cin>>students;
int *studentsList= new int[____];
for(int i =0; i<____;i++)
{
cin>>_____[i];
}
int* marksList = studentsList;

Print MarksList using for loop with students as iterations
return 0;
}
```

SUBMISSION GUIDELINES:

1. Complete your tasks.
2. Make sure each task is compiling and running successfully.
3. After execution, copy the code of every task in a single text file.
4. Submit the text file on Google Classroom

Your .txt file must look like this



```
//Task 1

#include<iostream>
using namespace std;

int main()
{
-----
}

//Task 2

#include<iostream>
using namespace std;

int main()
{
-----
}

//Task 3

#include<iostream>
using namespace std;

int main()
{
-----
}
```