Assignment-1

Problem Statement:

Write a program to find out the largest number using function.

Objective:

To make a function that can find the largest number.

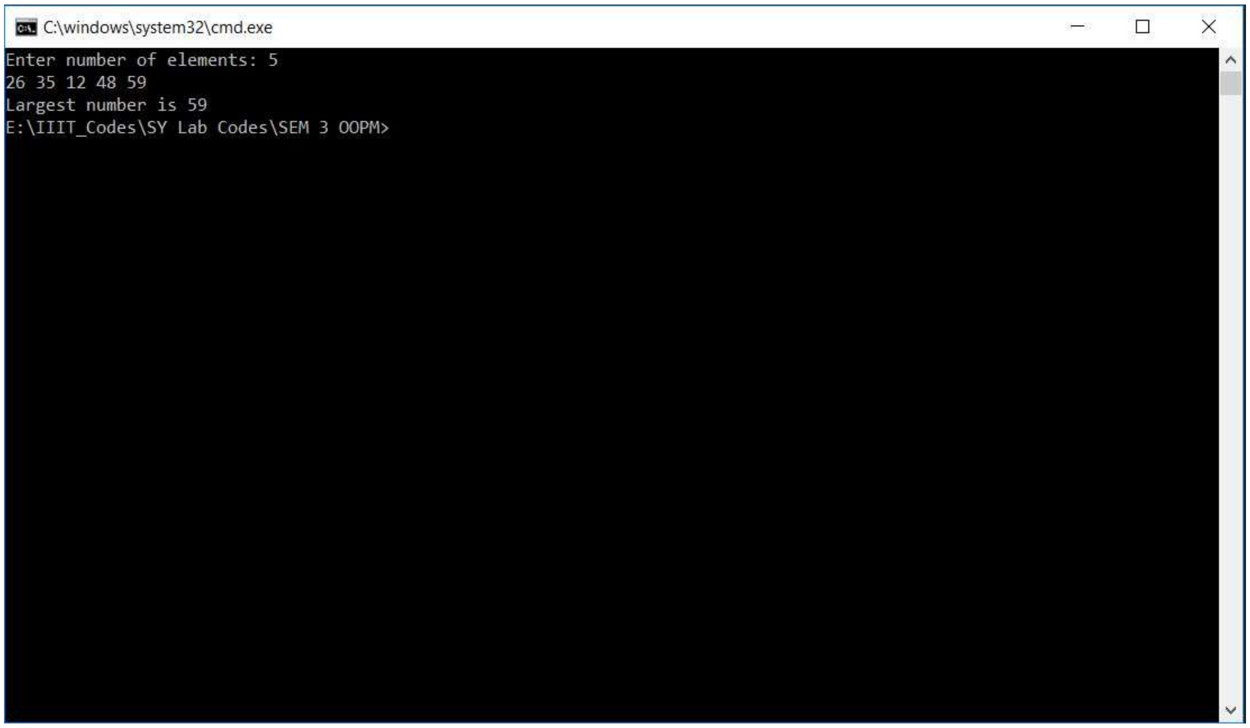
Theory:

A function is a set of statements that take inputs, do some specific computation and produces output. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

Code:

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

int largest(int arr[], int n)
{	int i; int max = arr[0];
	for (i = 1; i < n; i++)
		if (arr[i] > max)
			max = arr[i];
	return max;
}
int main()
{
	int arr[] = {10, 324, 45, 90, 9808}; int n;
	cout << "Enter number of elements: ";
	cin >> n;
	arr[n];
	for (int i = 0; i < n; i++)
	{
		int temp; cin >> temp; arr[i] = temp;
	}
	cout << "Largest number is " << largest(arr, n);
	return 0;
}
```

Output:


C:\windows\system32\cmd.exe

```
Enter number of elements: 5
26 35 12 48 59
Largest number is 59
E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-2

Problem Statement:

Write a program to find the area of circle, rectangle and triangle using function overloading.

Objective:

To make a function that can find the area.

Theory:

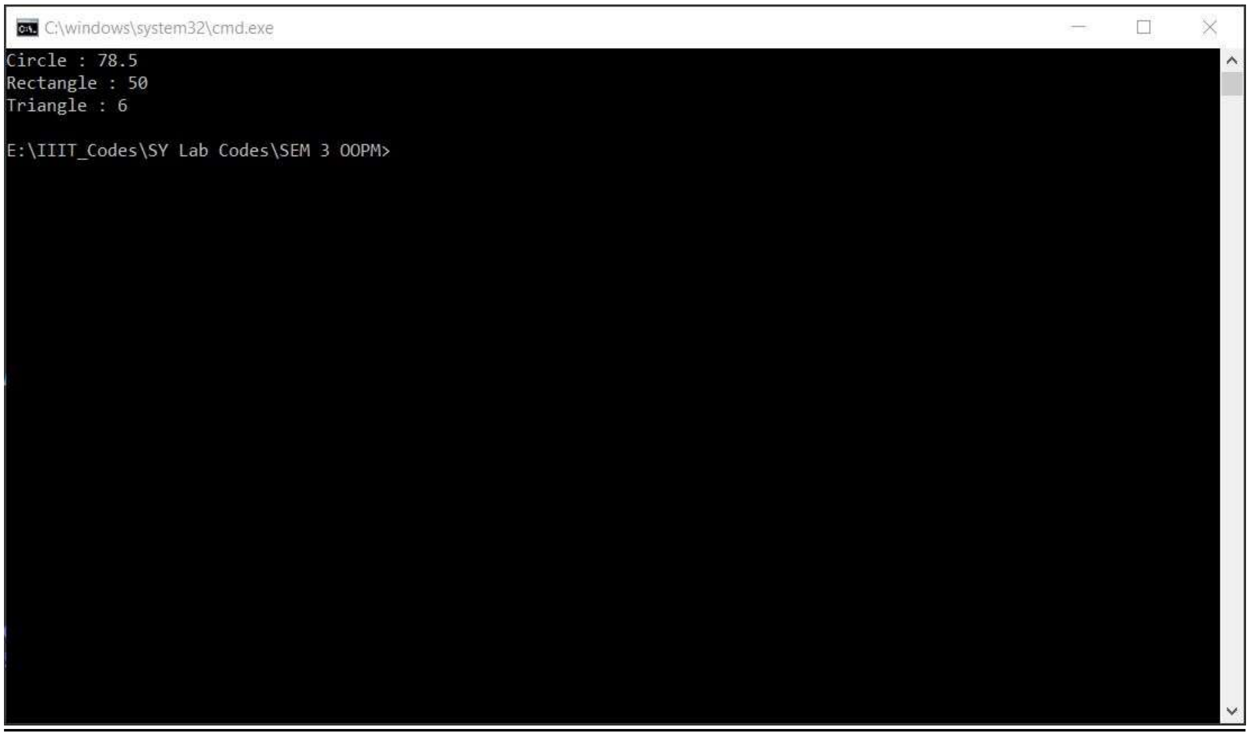
Function overloading-

Function overloading is a feature in c++ where two or more functions can have the same name but different parameters. It can be considered as an example of polymorphism feature in c++. We cannot overload function declarations that differ only by the return type. It is a compile time polymorphism.

Code:

```
#include<iostream>
#include<math.h>

using namespace std;

double area(double radius)
{
	return 3.14 * radius * radius;
}

double area(double l, double b)
{
	return l*b;
}

double area(double a, double b, double c)
{
	double s;
	s = (a+b+c)/2;
	return sqrt(s*(s-a)*(s-b)*(s-c));
}
```

```
int main()
{
	cout << "Circle : " << area(5.00) << "\n";
	cout << "Rectangle : " << area(5.00, 10.00) << "\n";
	cout << "Triangle : " << area(5.00,4.00,3.00) << "\n";
	return 0;
}
```
Output:



```
Circle : 78.5
Rectangle : 50
Triangle : 6

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-3

Problem Statement:

Write a program to implement complex numbers using operator overloading.

Objective:

To make a function that can be overloaded for implementation of complex number.

Theory

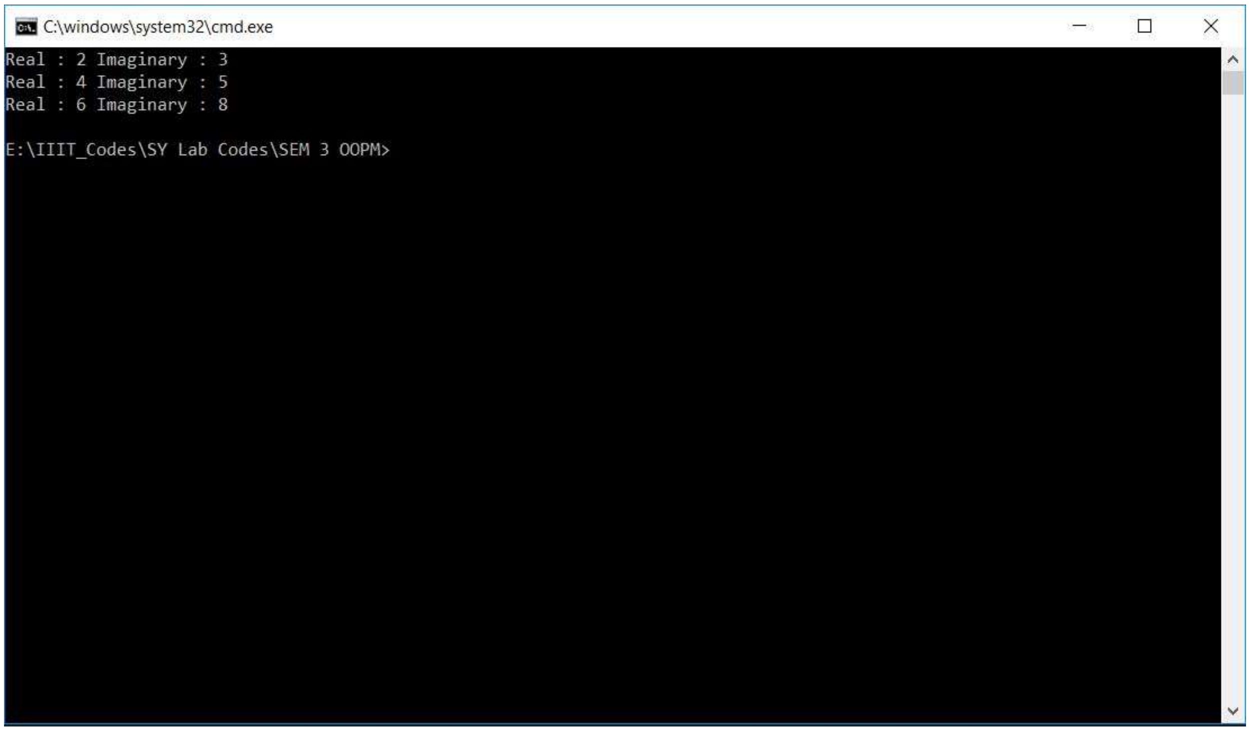
Operator overloading-

In c++ we can make operators to work for user defined classes. This means c++ has the ability to provide the operators with a special meaning for a data type,this ability is known as operator overloading. Operator overloading is a compile-time polymorphism.

Code:

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

class Complex
{
	int real,imag;

public:
	Complex()
	{

	}

	Complex(int r, int c)
	{
		real = r;
		imag = c;
	}

	Complex operator +(Complex c2);
	void Display()
	{
		cout << "Real : " << real << " Imaginary : " << imag << '\n';
	}
};

	Complex Complex::operator +(Complex c2)
	{
		Complex c;
		c.real = real + c2.real;
		c.imag = imag + c2.imag;
		return c;
	}

int main()
{
	Complex c1(2,3),c2(4,5),c3;
	c1.Display();
	c2.Display();
	c3 = c1+c2;
	c3.Display();
	return 0;
}
```

<u>Output:</u>



```
Real : 2 Imaginary : 3
Real : 4 Imaginary : 5
Real : 6 Imaginary : 8

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-4

Problem Statement:

Write a program using class and object to print bio-data of the students.

Objective:

To make class/object and to print bio-data of the students.

Theory

Class- The building block of c++ that leads to object oriented programming is a class. It is a user defined data type, which holds its own data members and member functions which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

Object- An object is an instance of a class when a class is defined, no memory is allocated but when it is instantiated memory is allocated.

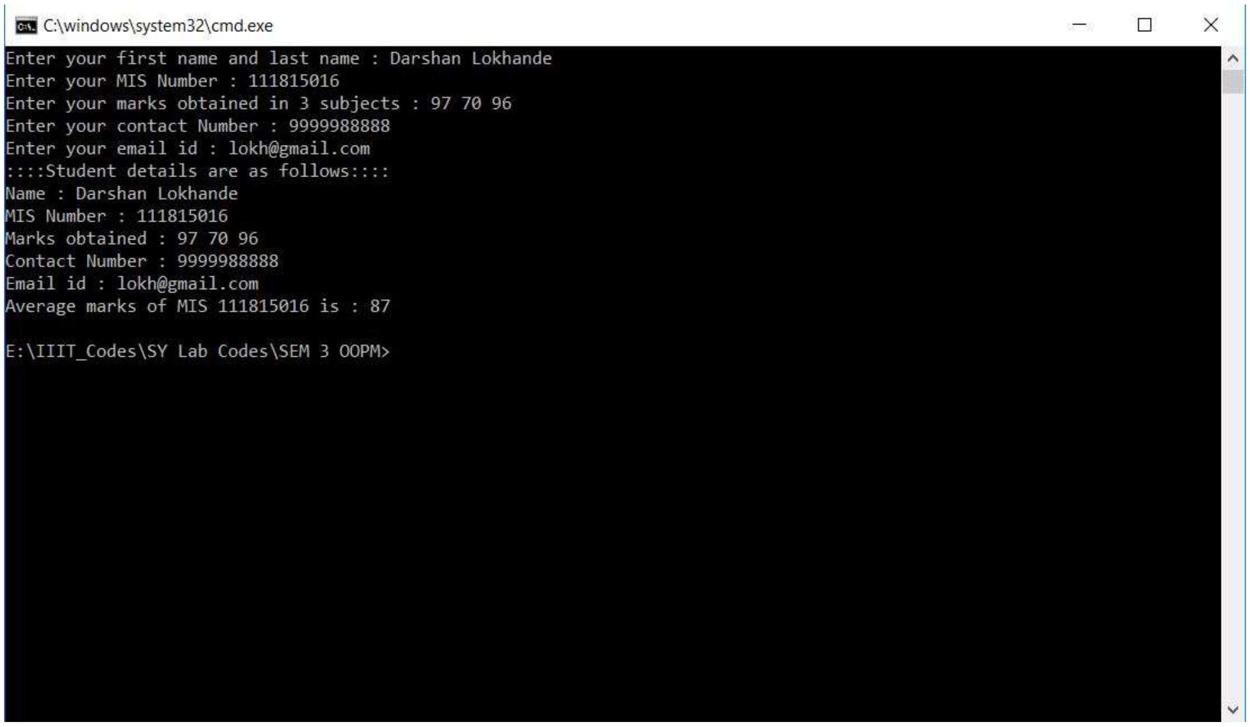
A class is defined in c++ using keyboard class followed by the name of class.

Code:

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

class Student
{
	string email,fname, lname;
	long long int m[3],num,MIS;
public:
	Student()
	{

	}
	void GetData()
	{
		cout << "Enter your first name and last name : ";
		cin >> fname >> lname;
		cout << "Enter your MIS Number : ";
		cin >> MIS;
		cout << "Enter your marks obtained in 3 subjects : ";
		cin >> m[0] >> m[1] >> m[2];
		cout << "Enter your contact Number : ";
		cin >> num;
		cout << "Enter your email id : ";
		cin >> email;
```

```
        }
        void Putdata()
        {
                cout << "::::Student details are as follows::::\n";
                cout << "Name : " << fname << ' ' << lname << '\n';
                cout << "MIS Number : " << MIS << '\n';
                cout << "Marks obtained : " << m[0] << ' ' << m[1] << ' ' <<  m[2] << '\n';
                cout << "Contact Number : " << num << '\n';
                cout << "Email id : " << email << '\n';

        }
        void Average()
        {
                cout << "Average marks of MIS " << MIS << " is : " << (m[0]+m[1]+m[2])/3 << '\n';
        }
};

int main()
{
        Student s1;
        s1.GetData();
        s1.Putdata();
        s1.Average();
        return 0;
}
```

Output:



```
Enter your first name and last name : Darshan Lokhande
Enter your MIS Number : 111815016
Enter your marks obtained in 3 subjects : 97 70 96
Enter your contact Number : 9999988888
Enter your email id : lokh@gmail.com
::::Student details are as follows::::
Name : Darshan Lokhande
MIS Number : 111815016
Marks obtained : 97 70 96
Contact Number : 9999988888
Email id : lokh@gmail.com
Average marks of MIS 111815016 is : 87

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-5

Problem Statement:

Write a program which defines a class with constructor and destructor which will count number of object created and destroyed.

Objective:

To make constructor and destructor that can find the number of objects.

Theory

Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to object members after storage is allocated to the object.

Whereas, **Destructor** on the other hand is used to destroy the class object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope.

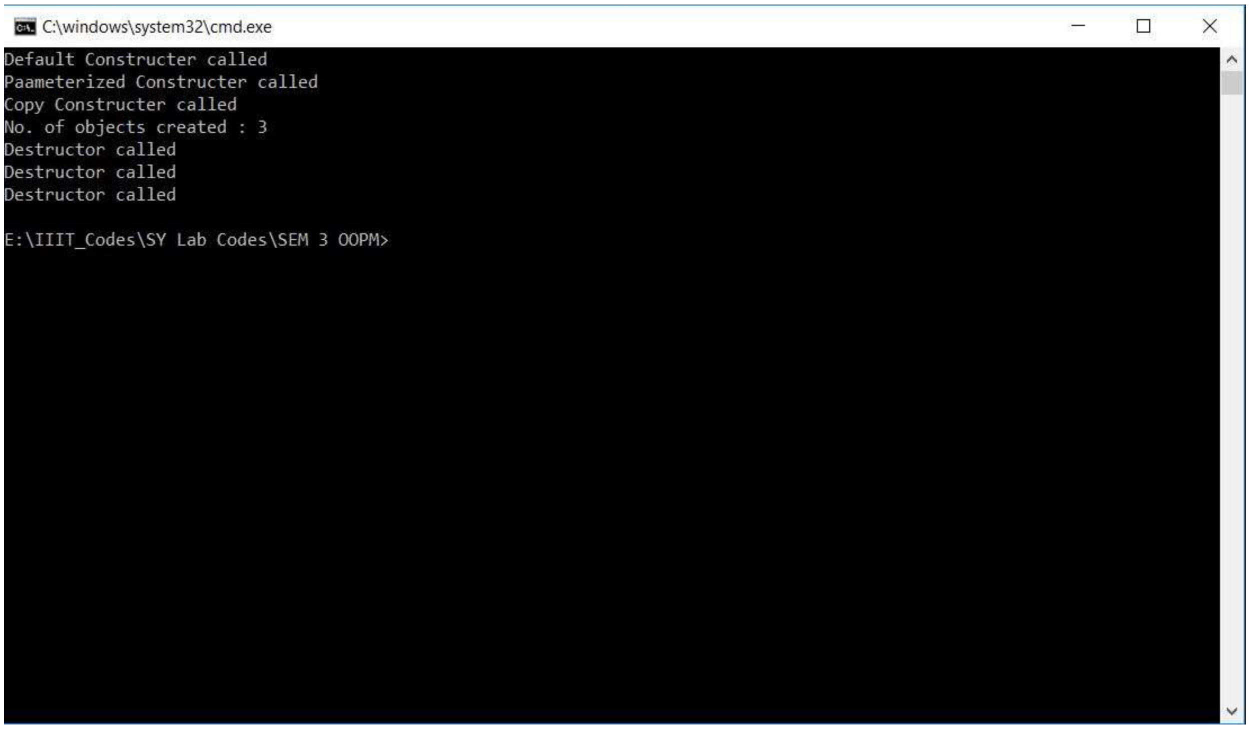
The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor; with a tilde ~ sign as prefix to it.

Code:

```
#include<iostream>

using namespace std;

int objc = 0;
int objd = 0;

class Darshan
{
public:
	Darshan()
	{
		cout << "Default Constructer called\n";
		objc++;
	}
	Darshan(int a)
	{
		cout << "Paameterized Constructer called\n";
		objc++;
	}
	Darshan(Darshan &d)
```

```
        {

                cout << "Copy Constructer called\n";
                objc++;
        }
        ~Darshan()
        {
                cout << "Destructor called\n";
                objd++;
        }
};

int main()
{
        Darshan d1,d2(68);
        Darshan d3(d2);

        cout << "No. of objects created : " << objc << endl;
        // cout << "No. of objects Destructed : " << objd <<endl;
        return 0;
}
```

Output:



```
Default Constructer called
Paameterized Constructer called
Copy Constructer called
No. of objects created : 3
Destructor called
Destructor called
Destructor called

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-6

Problem Statement:

Write a program to show applications of different types of inheritances.

Objective:

To make a class that can be used to show applications of different types of inheritances.

Theory

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object Oriented Programming.

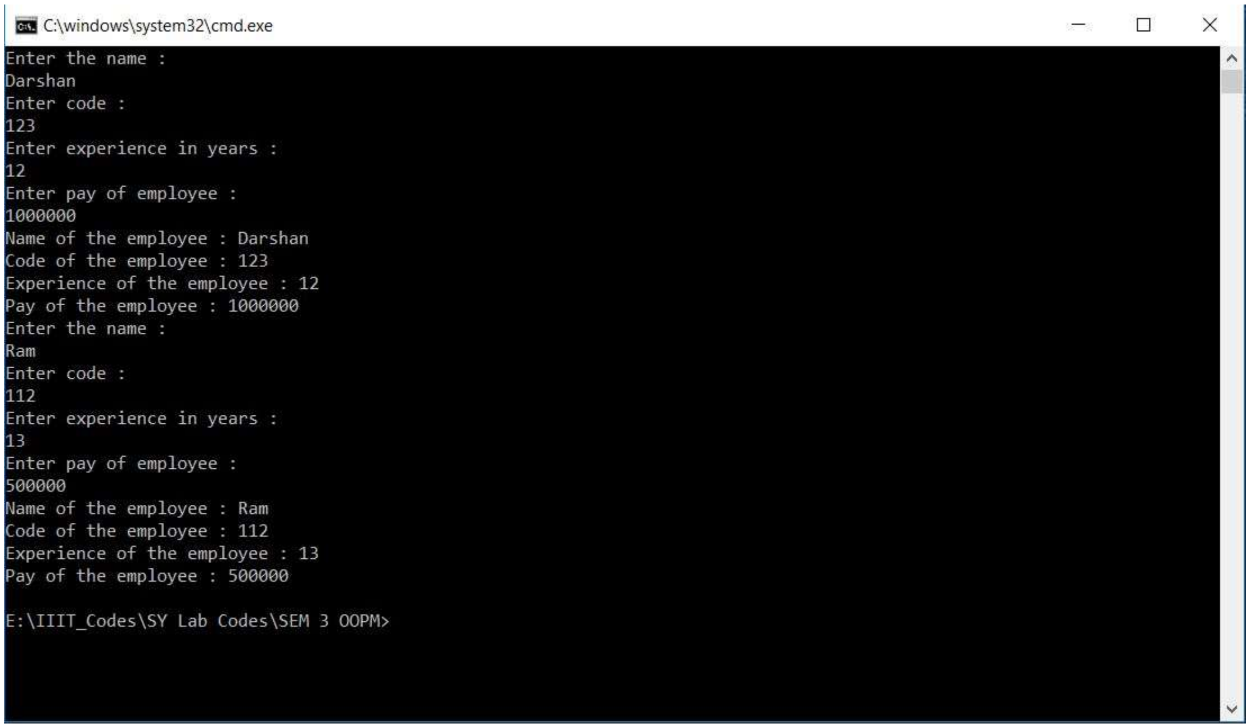
Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.

Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.

Code:

```
#include<bits/stdc++.h>

using namespace std;
class person
{
protected:
        int code;
public:
        string name;
        void getdata()
        {
                cout << "Enter the name : " << '\n';
                cin >> name;
                cout << "Enter code : " << '\n';
                cin >> code;
        }
};
class admin: virtual public person
{
protected:
        int exp;
public:
        void input()
        {
                cout << "Enter experience in years : " << '\n';
                cin >> exp;
        }
};
class account: virtual public person
{
protected:
        int pay;
public:
        void get()
        {
                cout << "Enter pay of employee : " << '\n';
                cin >> pay;
        }
};
class master: public admin, public account
{
public:
        void showdata()
        {
                cout << "Name of the employee : " << name << '\n';
                cout << "Code of the employee : " << code << '\n';
                cout << "Experience of the employee : " << exp << '\n';
                cout << "Pay of the employee : " << pay << '\n';
        }
};
int main()
{
        master m1, m2;
        m1.getdata();
        m1.input();
        m1.get();
        m1.showdata();

        m2.getdata();
        m2.input();
        m2.get();
        m2.showdata();

        return 0;
}
```

Output:



```
Enter the name :
Darshan
Enter code :
123
Enter experience in years :
12
Enter pay of employee :
1000000
Name of the employee : Darshan
Code of the employee : 123
Experience of the employee : 12
Pay of the employee : 1000000
Enter the name :
Ram
Enter code :
112
Enter experience in years :
13
Enter pay of employee :
500000
Name of the employee : Ram
Code of the employee : 112
Experience of the employee : 13
Pay of the employee : 500000

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-7

Problem Statement:

Write a program to add two private data members using friend function.

Objective:

To make a friend function that can find the sum of two private data members.

Theory

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

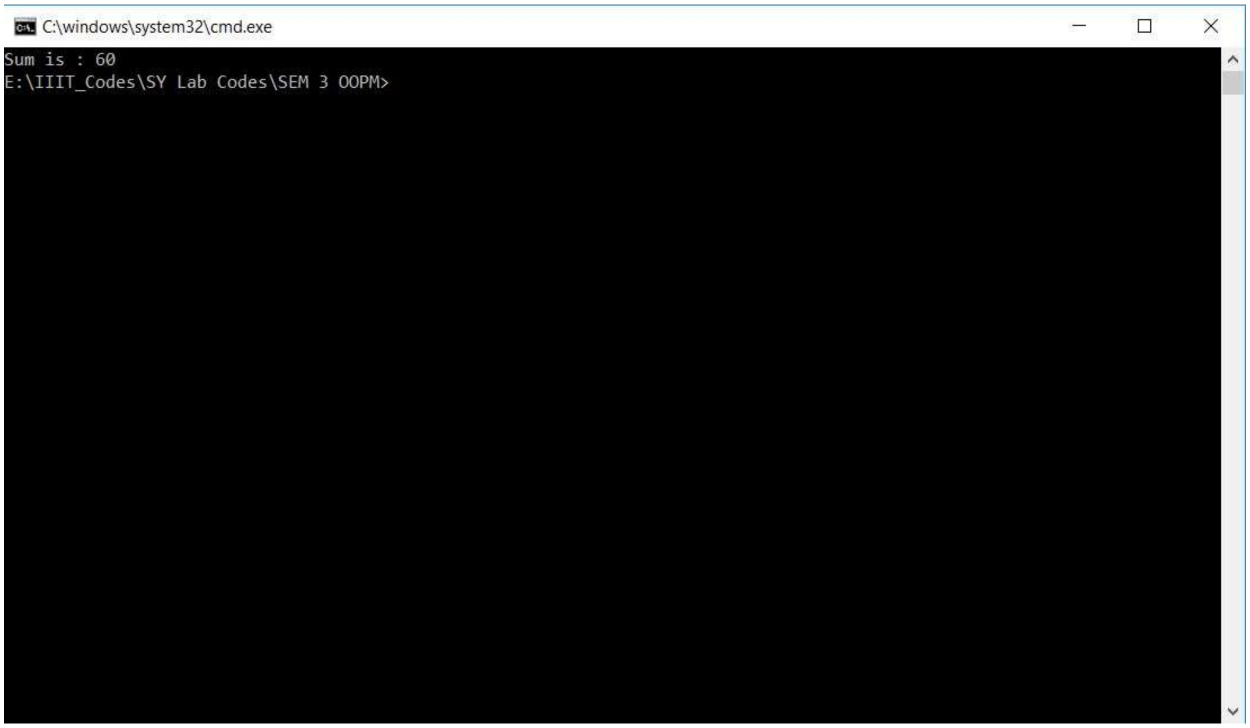
A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.

Code:

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

class Darshan
{
	int a,b;
public:
	void GetData(int x, int y)
	{
		a = x;
		b = y;
	}
	friend void add(Darshan);
};

void add (Darshan d1)
{
	int sum;
	sum = d1.a + d1.b;
	cout << "Sum is : " << sum;

}

int main()
{
	Darshan d1,d2;
	d1.GetData(10,50);
```

```
        d2.GetData(50,68);
        add(d1);
        return 0;
}
```

<u>Output:</u>


C:\windows\system32\cmd.exe

```
Sum is : 60
E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-8

Problem Statement:

Write a program using dynamic memory allocation to perform 2x2 matrix addition and subtraction.

Objective:

To make matrix using dynamic memory allocation and perform 2x2 matrix addition and subtraction.

Theory

Dynamic memory allocation in C/C++ refers to performing memory allocation manually by programmer. Dynamically allocated memory is allocated on Heap and non-static and local variables get memory allocated on Stack.

The new operator denotes a request for memory allocation on the Heap. If sufficient memory is available, new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable.

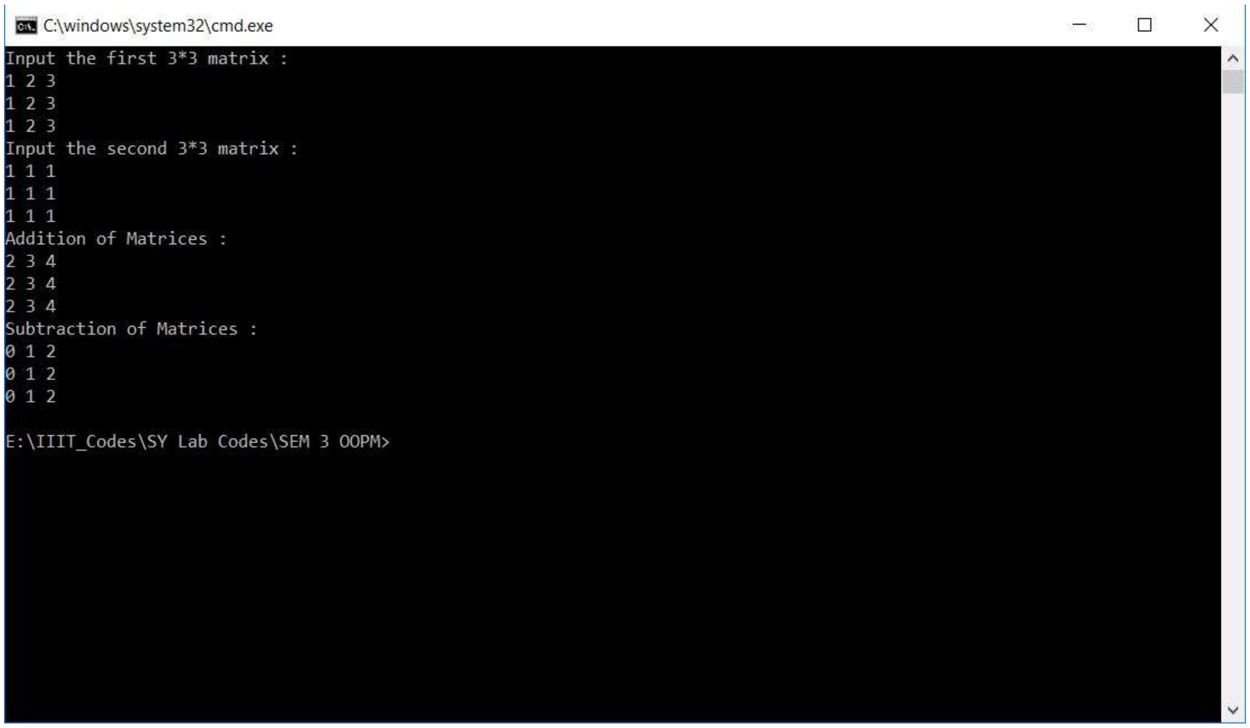
Since it is programmer's responsibility to de-allocate dynamically allocated memory, programmers are provided delete operator by C++ language.

Code:

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

int main()
{
	int **p[4];
	int i,j;
	for(i=0;i<4;i++)
	{
		p[i] = new int*[3];
		for(j=0;j<3;j++)
		{
			p[i][j] = new int [3];
		}
	}
	cout << "Input the first 3*3 matrix :\n";
	for(i=0;i<3;i++)
	{
		for(j=0;j<3;j++)
		{
```

```
                        cin >> p[0][i][j];
                }
        }
        cout << "Input the second 3*3 matrix :\n";
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        cin >> p[1][i][j];
                }
        }
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        p[2][i][j] = p[0][i][j] + p[1][i][j];
                        p[3][i][j] = p[0][i][j] - p[1][i][j];
                }
        }
        cout << "Addition of Matrices : \n";
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                        cout << p[2][i][j] << ' ';
                cout << '\n';
        }
        cout << "Subtraction of Matrices : \n";
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                        cout << p[3][i][j] << ' ';
                cout << '\n';
        }
        return 0;
}
```

Output:



```
Input the first 3*3 matrix :
1 2 3
1 2 3
1 2 3
Input the second 3*3 matrix :
1 1 1
1 1 1
1 1 1
Addition of Matrices :
2 3 4
2 3 4
2 3 4
Subtraction of Matrices :
0 1 2
0 1 2
0 1 2

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-9

Problem Statement:

Write a program to show the application of virtual function.

Objective:

To make a virtual function that can be used to solve ambiguity problem.

Theory

A virtual function is a member function which is declared within a base class and is re-defined (Overriden) by a derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.

Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.

They are mainly used to achieve Runtime polymorphism

Functions are declared with a virtual keyword in base class.

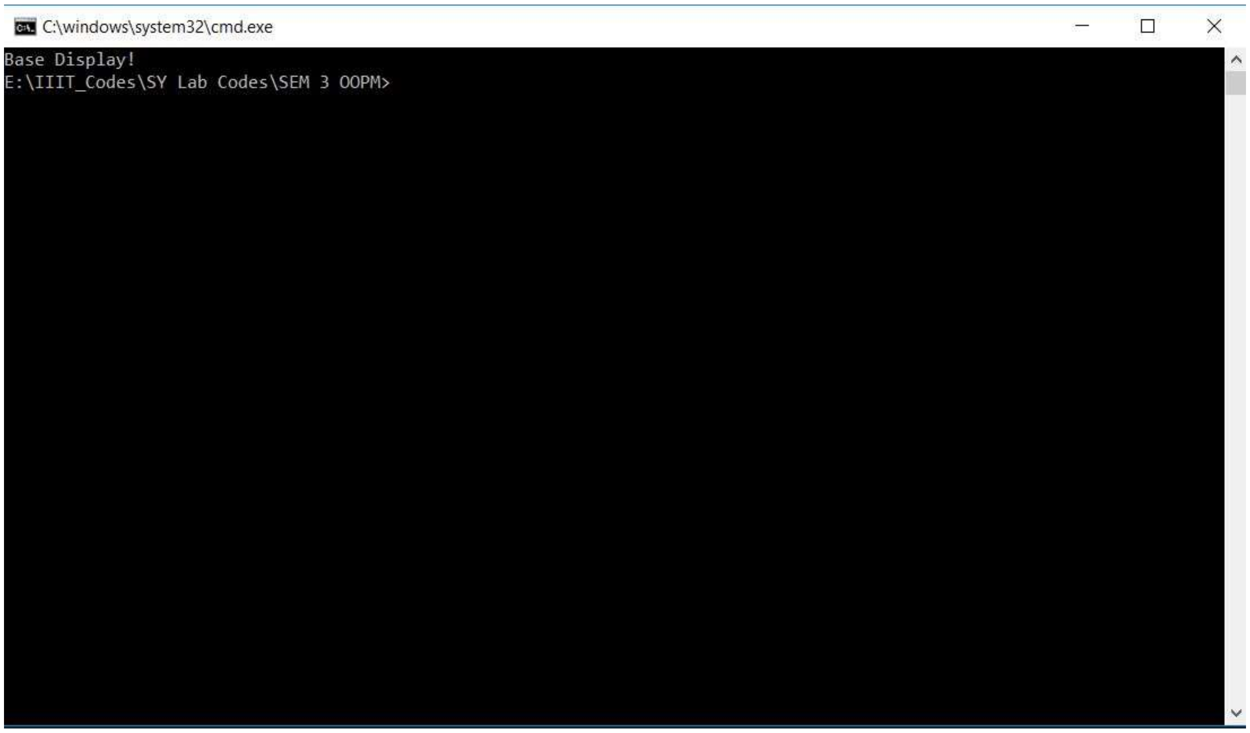
The resolving of function call is done at Run-time.

Code:

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

class A
{
public:
	virtual void Display()
	{
		cout << "Base Display!";
	}
};

class B:public A
{
public:
	void show()
	{
		cout << "Derived Display!";
	}
};
```

```
int main()
{
        A *p;
        B b;
        p = &b;
        p ->Display();
        return 0;
}
```

<u>Output:</u>



```
Base Display!
E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-10

Problem Statement:

Write a program that store five student records in a file.

Objective:

To make a file handling program to store record of five students.

Theory

Files are used to store data in a storage device permanently. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in the fstream header-file.

ofstream: Stream class to write on files

ifstream: Stream class to read from files

fstream: Stream class to both read and write from/to files.

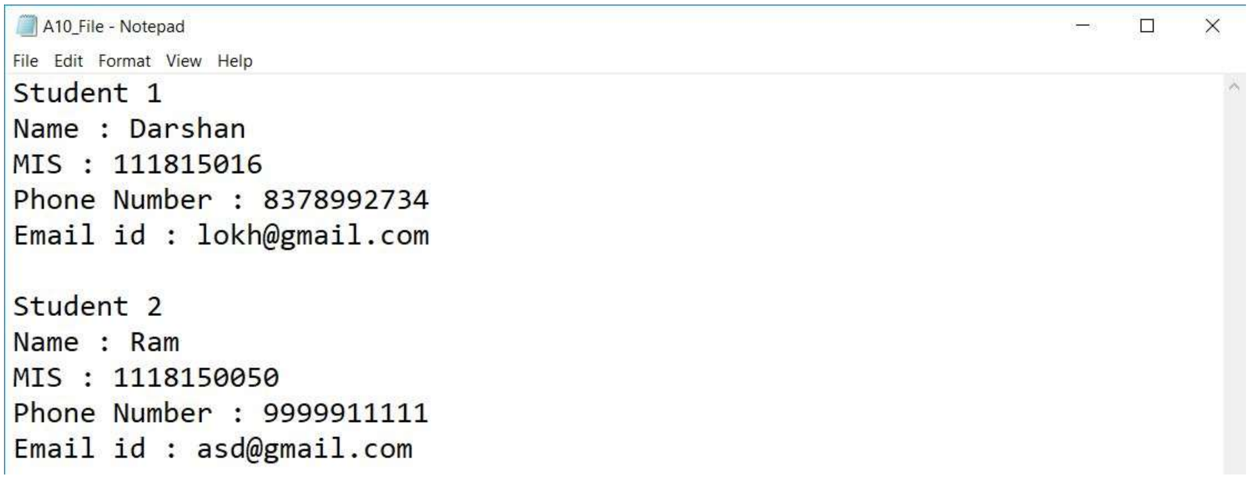
Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
	ofstream file;
	file.open("A10_File.txt");
	int n;
	string str;
	cout << "Enter the number of students : ";
	cin >> n;

	for(int i=0 ; i<n ; i++)
	{
		cout << "Enter the Details of student " << i+1 << " :\n";
		file << "Student " << i+1 << '\n';
		cout << "Name : ";
		cin >> str;
		file << "Name : " << str << '\n';
		cout << "MIS : ";
		cin >> str;
		file << "MIS : " << str << '\n';
		cout << "Phone Number : ";
```

```
                cin >> str;
                file << "Phone Number : " << str << '\n';
                cout << "Email id : ";
                cin >> str;
                file << "Email id : " << str << "\n\n";
        }
        file.close();
        return 0;
}
```

Output:

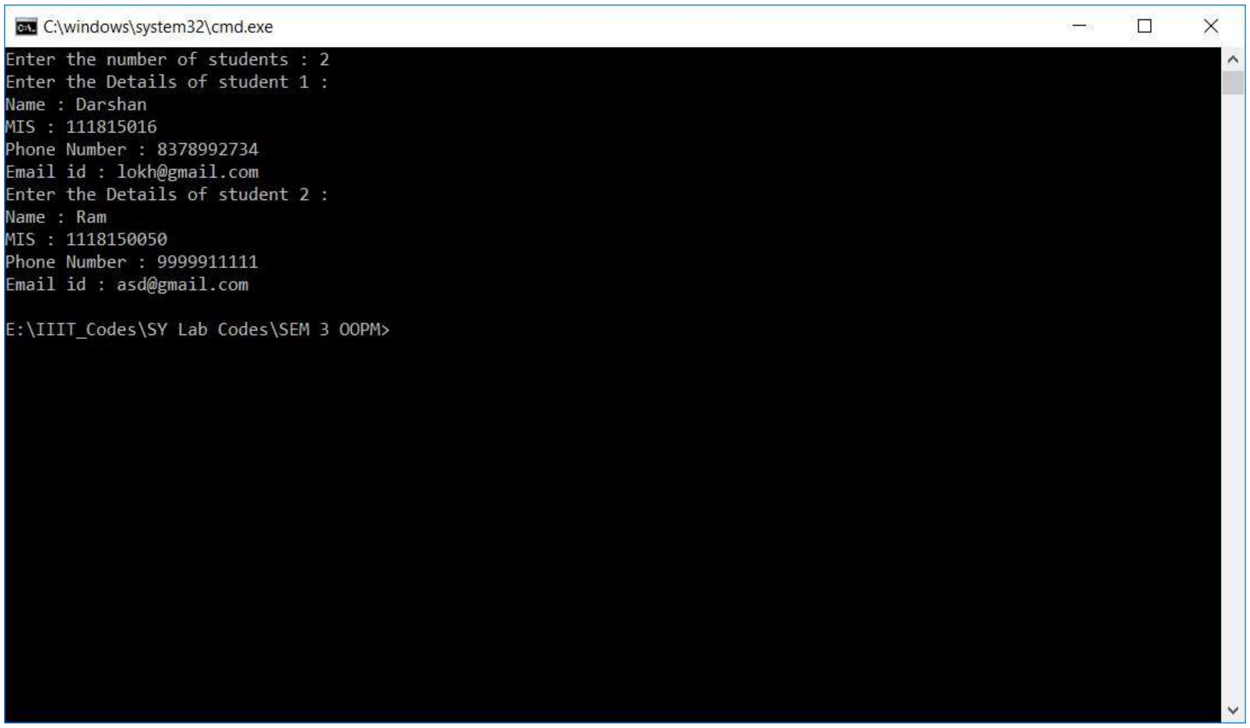
Text file "Student.txt:



A10_File - Notepad

```
Student 1
Name : Darshan
MIS : 111815016
Phone Number : 8378992734
Email id : lokh@gmail.com

Student 2
Name : Ram
MIS : 1118150050
Phone Number : 9999911111
Email id : asd@gmail.com
```

Output:



C:\windows\system32\cmd.exe

```
Enter the number of students : 2
Enter the Details of student 1 :
Name : Darshan
MIS : 111815016
Phone Number : 8378992734
Email id : lokh@gmail.com
Enter the Details of student 2 :
Name : Ram
MIS : 1118150050
Phone Number : 9999911111
Email id : asd@gmail.com

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-11

Problem Statement:

Write a program to show the application of class/function template.

Objective:

To make a function or class template that can operate with generic types.

Theory

Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.

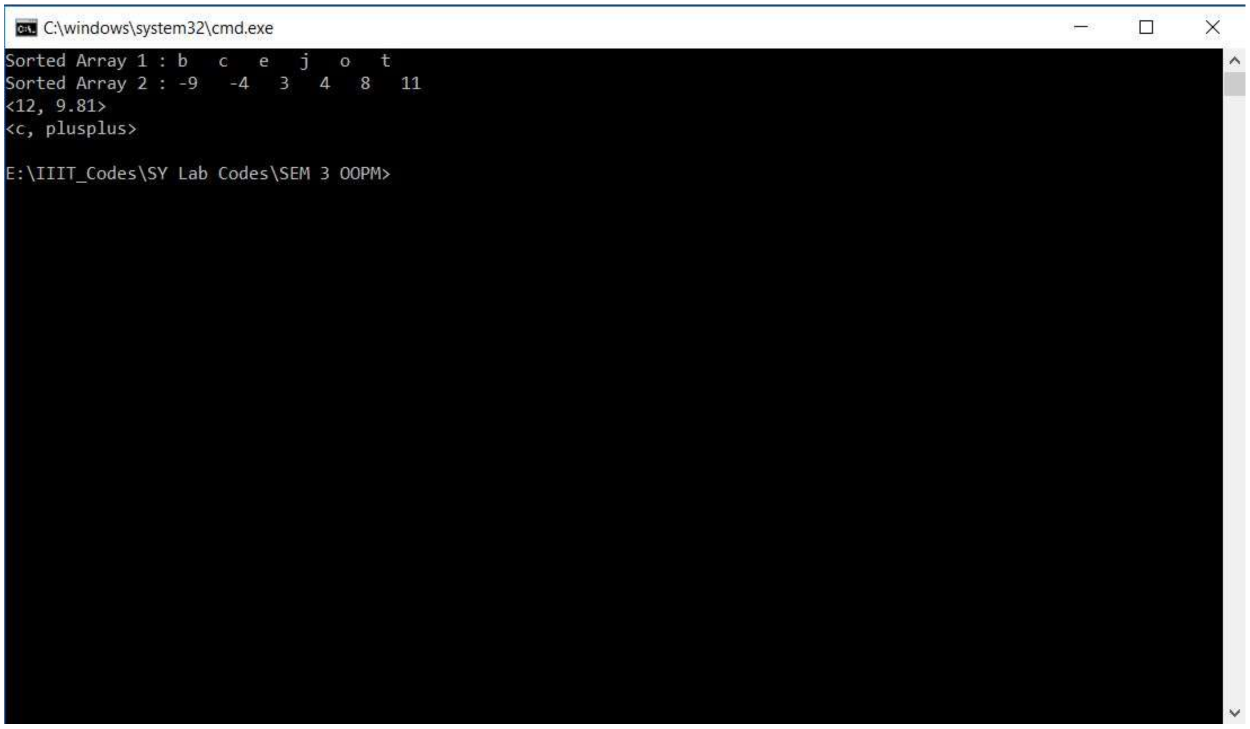
In C++ this can be achieved using template parameters. A template parameter is a special kind of parameter that can be used to pass a type as argument: just like regular function parameters can be used to pass values to a function, template parameters allow to pass also types to a function. These function templates can use these parameters as if they were any other regular type.

Code:

```
#include <bits/stdc++.h>

using namespace std;

template <class T>
void BubbleSort(T arr[], int n)
{
	int i, j, flag;
	T temp;

	for (i = 0; i < n - 1; i++)
	{
		flag = 0;
		for (j = 0; j < n - 1 - i; j++)
		{
			if (arr[j] > arr[j + 1])
			{
				flag = 1;
				temp = arr[j];
				arr[j] = arr[j + 1];
				arr[j + 1] = temp;
			}
		}
```

```
                if (flag == 0)
                {
                        break;
                }
        }
}

template <class T>
void printArray(T a[], int n)
{
        for (size_t i = 0; i < n; ++i)
                cout << a[i] << "   ";
        cout << '\n';
}

template <class T1 , class T2>

class tempclass
{
        T1 p;
        T2 q;
public:
        tempclass(T1 a, T2 b)
        {
                p = a;
                q = b;
        }
        void show()
        {
                cout << "<" << p << ", " << q << ">" << '\n';
        }
};

int main()
{
        char array1[] = {'o', 'b', 'j', 'e', 'c', 't'};
        int array2[] = {3, -9, 8, 4, 11, -4};
        int n1, n2;
        n1 = sizeof(array1) / sizeof(array1[0]);
        n2 = sizeof(array2) / sizeof(array2[0]);
        BubbleSort (array1, n1);
        BubbleSort (array2, n2);
        cout << "Sorted Array 1 : ";
        printArray(array1, n1);
        cout << "Sorted Array 2 : ";
        printArray(array2, n2);

        tempclass <int, float> a(12, 9.81);
        tempclass <char, string> b('c', "plusplus");
        a.show();
        b.show();

        return 0;
}
```

Output:



C:\windows\system32\cmd.exe

```
Sorted Array 1 : b   c   e   j   o   t
Sorted Array 2 : -9   -4   3   4   8   11
<12, 9.81>
<c, plusplus>

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```

Assignment-12

Problem Statement:

Write a program to show the application of exception handling.

Objective:

To make a perfect program so that the user will not get any kind of error.

Theory

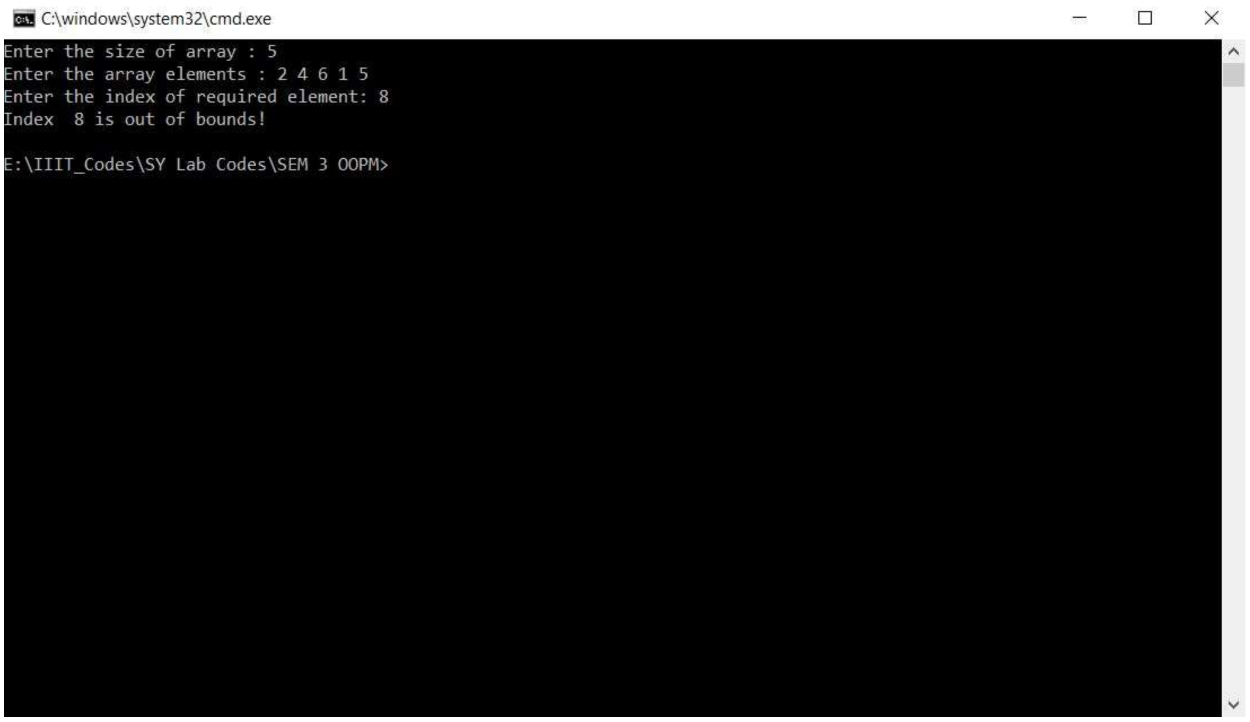
An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.

Code:

```
#include<bits/stdc++.h>
using namespace std;

int main() {
	int n;
	cout << "Enter the size of array : ";
	cin >> n;
	int arr[n];
	cout << "Enter the array elements : ";
	for (int i = 0; i < n; i++)
		cin >> arr[i];
	int index;
	cout << "Enter the index of required element: ";
	cin >> index;
	try {
		if (index > n - 1) {
			throw - 1;
		}
		else {
			cout << "Element at index  " << index << " is: " << arr[index];
		}
	}
	catch (int) {
		cout << "Index  " << index << " is out of bounds!" << endl;
	}
	return 0;
}
```

Output:



C:\windows\system32\cmd.exe

```
Enter the size of array : 5
Enter the array elements : 2 4 6 1 5
Enter the index of required element: 8
Index  8 is out of bounds!

E:\IIIT_Codes\SY Lab Codes\SEM 3 OOPM>
```