Lab Questions

1. Write a program to define a class that uses static data members and static member functions to count and display the number of objects created.

```
#include<iostream>
using namespace std;
class Counter{
private:
   static int count;
public:
   Counter(){
   count++;
   cout<<"Constructor called. Object number:"<<count<<endl;
   }
   static void displaycount()
  {
     cout<<"Total Number of Objects Created:"<<endl;
  }
};
  int Counter::count=0;
  int main()
  {
     cout<<"--Object Creation--"<<endl;
     Counter obj1;
     Counter obj2;
     Counter obj3;
     cout<<"Display Count Using Static Function--"<<endl;
     Counter::displaycount();
     return 0;
  }
```

Output:

```
--Object Creation--
Constructor called. Object number:1
Constructor called. Object number:2
Constructor called. Object number:3
Display Count Using Static Function--
Total Number of Objects Created:
```

2. Write a program to create a class that uses a copy constructor to copy data from one object to another.

```
#include <iostream>
using namespace std;
class Student {
private:
int roll;
string name;
public:
Student(int r, string n) {
roll = r;
name = n;
cout << "Parameterized constructor called." << endl;
}
Student(const Student &s) {
roll = s.roll;
name = s.name;
cout << "Copy constructor called." << endl;
}
void display() {
cout << "Name: " << name << ", Roll Number: " << roll << endl;
}
};
int main() {
int r;
string n;
```

```
cout << "Enter student name: ";
getline(cin, n);
cout << "Enter roll number: ";
cin >> r;
cout << "\n--- Creating Original Object ---" << endl;
Student s1(r, n);
cout << "\n--- Creating Copy Using Copy Constructor ---" << endl;
Student s2 = s1;
cout << "\n--- Displaying Objects ---" << endl;
cout << "Original Object: ";
s1.display();
cout << "Copied Object: ";
s2.display();
return 0;
}
```

Output:

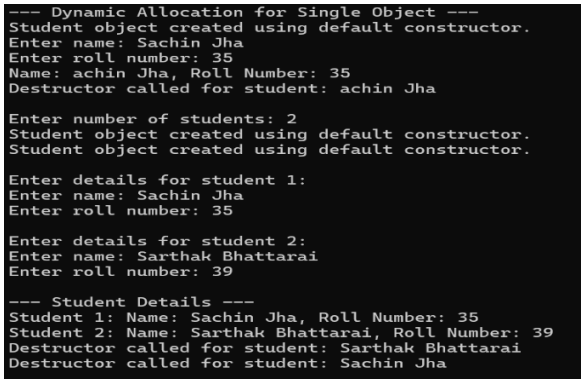
```
Enter student name: Sachin Jha
Enter roll number: 35

--- Creating Original Object ---
Parameterized constructor called.

--- Creating Copy Using Copy Constructor ---
Copy constructor called.

--- Displaying Objects ---
Original Object: Name: Sachin Jha, Roll Number: 35
Copied Object: Name: Sachin Jha, Roll Number: 35
```

3. Write a program to dynamically allocate and deallocate memory for a single object and an array of objects using the new and delete operators.

```
#include <iostream>
using namespace std;
class Student {
private:
string name;
int roll;
public:
Student() {
name = "";
roll = 0;
cout << "Student object created using default constructor." << endl;
}
Student(string n, int r) {
name = n;
roll = r;
cout << "Student object created using parameterized constructor." << endl;
}
void input() {
cout << "Enter name: ";
cin.ignore();
getline(cin, name);
cout << "Enter roll number: ";
cin >> roll;
}
void display() {
cout << "Name: " << name << ", Roll Number: " << roll << endl;
}
~Student() {
cout << "Destructor called for student: " << name << endl;
}
```

```
};
int main() {
cout << "--- Dynamic Allocation for Single Object ---" << endl;
Student *s1 = new Student;
s1->input();
s1->display();
delete s1;
int n;
cout << "\nEnter number of students: ";
cin >> n;
Student* sArray = new Student[n];
for (int i = 0; i < n; i++) {
cout << "\nEnter details for student " << i + 1 << ":" << endl;
sArray[i].input();
}
cout << "\n--- Student Details ---" << endl;
for (int i = 0; i < n; i++) {
cout << "Student " << i + 1 << ": ";
sArray[i].display();
}
delete[] sArray;
return 0;
}
```

Output:



4. Write a program that demonstrates default, parameterized, and copy constructors. Use the this pointer to calculate the midpoint between two points.

```
#include <iostream>
using namespace std;
class Point {
private:
float x, y;
public:
// Default constructor
Point() {
x = 0;
y = 0;
cout << "Default constructor called.\n";
}
// Parameterized constructor
Point(float a, float b) {
x = a;
y = b;
cout << "Parameterized constructor called for (" << x << ", " << y << ")\n";
}
// Copy constructor
Point(const Point &p) {
x = p.x;
y = p.y;
cout << "Copy constructor called for (" << x << ", " << y << ")\n";
}
// Function to calculate midpoint using this pointer
Point midpoint(const Point &p) const {
float midX = (this->x + p.x) / 2;
float midY = (this->y + p.y) / 2;
return Point(midX, midY); // returning new object
}
```

```
void input() {
cout << "Enter x and y coordinates: ";
cin >> x >> y;
}
void display() const {
cout << "(" << x << ", " << y << ")" << endl;
}
};
int main() {
cout << "--- Enter Coordinates for Point 1 ---" << endl;
Point p1;
p1.input();
cout << "--- Enter Coordinates for Point 2 ---" << endl;
Point p2;
p2.input();
cout << "--- Copying Point 1 to Point 3 ---" << endl;
Point p3 = p1; // invokes copy constructor
cout << "\n--- Midpoint of Point 1 and Point 2 ---" << endl;
Point mid = p1.midpoint(p2); // uses this pointer
cout << "\nPoint 1: "; p1.display();
cout << "Point 2: "; p2.display();
cout << "Copied Point (p3): "; p3.display();
cout << "Midpoint: "; mid.display();
return 0;
}
```

Output:

```
--- Enter Coordinates for Point 1 ---
Default constructor called.
Enter x and y coordinates: 2
4
--- Enter Coordinates for Point 2 ---
Default constructor called.
Enter x and y coordinates: 3
5
--- Copying Point 1 to Point 3 ---
Copy constructor called for (2, 4)

--- Midpoint of Point 1 and Point 2 ---
Parameterized constructor called for (2.5, 4.5)

Point 1: (2, 4)
Point 2: (3, 5)
Copied Point (p3): (2, 4)
Midpoint: (2.5, 4.5)
```

5. Write a program to define a function that takes an object as a reference parameter and modifies its data members.

```
#include <iostream>

using namespace std;

namespace Office{

class Employee {

private:

string name;

int roll;

public:

Student() {

name = "";

roll = 0;

}

void input() {

cout << "Enter name: ";

getline(cin, name);

cout << "Enter roll number: ";
```

```
cin >> roll;
cin.ignore();
}
void display() const {
cout << "Name: " << name << ", Roll: " << roll << endl;
}
friend void modify(Employee& e);
};
}
void modify(Employee& e) {
cout << "\n--- Modifying Employee Details ---" << endl;
cout << "Enter new name: ";
getline(cin, e.name);
cout << "Enter new salary: ";
cin >> e.salary;
}
int main() {
Employee e;
cout << "------ Enter Initial Employee Data ------" << endl;
e.input();
cout << "\n----- Before Modification -----" << endl;
e.display();
modify(e);
cout << "\n----- After Modification -----" << endl;
e.display();
return 0;
}
```

Output:

```
------ Enter Initial Student Data ------
Enter name: Sachin Jha
Enter roll number: 35

----- Before Modification -----
Name: Sachin Jha, Roll: 35

--- Modifying Student Details ---
Enter new name: Sarthak Bhattarai
Enter new roll number: 39

----- After Modification -----
Name: Sarthak Bhattarai, Roll: 39
```

6. Write a program that defines a class inside a namespace and uses a reference to modify object attributes.

```
#include <iostream>

using namespace std;

namespace Office {

class Employee {

private:

string name;

float salary;

public:

Employee() {

name = "";

salary = 0;

}

void input() {

cout << "Enter Empoloyee name: ";

getline(cin, name);

cout << "Enter salary: ";

cin >> salary;

cin.ignore();

}
```

```
void display() const {
cout << "Name: " << name << ", Salary: " << salary << endl;
}
friend void updateEmployee(Office::Employee& e);
};
void updateEmployee(Office::Employee& e) {
cout << "\n--- Modifying Employee Data ---\n";
cout << "Enter new name: ";
getline(cin, e.name);
cout << "Enter new salary: ";
cin >>e.salary;
}
}
int main() {
Office::Employee e;
cout << "--- Enter Employee Data ---" << endl;
e.input();
cout << "\n--- Before Modification ---" << endl;
e.display();
updateEmployee(e);
cout << "\n--- After Modification ---" << endl;
e.display();
return 0;
}
```

Output:

```
--- Enter Employee Data ---
Enter Empoloyee name: Sachin Jha
Enter salary: 350000

--- Before Modification ---
Name: Sachin Jha, Salary: 350000

--- Modifying Employee Data ---
Enter new name: Sarthak Bhattarai
Enter new salary: 390000

--- After Modification ---
Name: Sarthak Bhattarai, Salary: 390000
```

7. Write a program that defines a constant member function to access constant object data, and use const_cast to modify it safely.

```
#include <iostream>
using namespace std;
class Student {
private:
string name;
int roll;
public:
Student(string n = "N/A", int r = 0) {
name = n;
roll = r;
}
void display() const {
cout << "Name: " << name << ", Roll: " << roll << endl;
}
void modify() const {
cout << "\n--- Attempting to Modify Constant Object ---" << endl;
Student* modifiable = const_cast<Student*>(this);
cout << "Enter new name: ";
getline(cin, modifiable->name);
cout << "Enter new roll number: ";
cin >> modifiable->roll;
cin.ignore();
}
};
int main() {
const Student s("Original", 101);
cout << "--- Constant Object (Before Modification) ---" << endl;
s.display();
s.modify();
cout << "\n--- Constant Object (After Modification) ---" << endl;
```

s.display();

return 0;

}

Output:

```
--- Constant Object (Before Modification) ---
Name: Original, Roll: 101

--- Attempting to Modify Constant Object ---
Enter new name: Rohit Sharma
Enter new roll number: 45

--- Constant Object (After Modification) ---
Name: Rohit Sharma, Roll: 45
```

8. Write a program to demonstrate a friend function that accesses private data from two different classes and adds their values.

#include <iostream>

using namespace std;

class ClassB;

class ClassA {

private:

int valueA;

public:

ClassA(int a = 0) {

valueA = a;

}

friend int addValues(ClassA, ClassB);

};

class ClassB {

private:

int valueB;

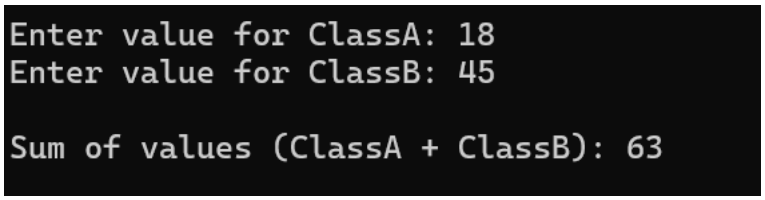
public:

ClassB(int b = 0) {

valueB = b;

```
}
friend int addValues(ClassA, ClassB);
};
int addValues(ClassA a, ClassB b) {
return a.valueA + b.valueB;
}
int main() {
int x, y;
cout << "Enter value for ClassA: ";
cin >> x;
cout << "Enter value for ClassB: ";
cin >> y;
ClassA objA(x);
ClassB objB(y);
int result = addValues(objA, objB);
cout << "\nSum of values (ClassA + ClassB): " << result << endl;
return 0;
}
```

Output:



9. Write a program to define a class String that uses a dynamic constructor to allocate memory and join two strings entered by the user.

```
#include <iostream>
#include <cstring>
using namespace std;
class MyString {
private:
char* str;
public:
MyString(const char* s1, const char* s2) {
int len = strlen(s1) + strlen(s2);
str = new char[len + 1]; // +1 for null terminator
strcpy(str, s1);
strcat(str, s2);
cout << "Dynamic constructor called. Strings joined." << endl;
}
void display() const {
cout << "Joined String: " << str << endl;
}
~MyString() {
delete[] str;
cout << "Destructor called. Memory released." << endl;
}
};
int main() {
char input1[100], input2[100];
cout << "Enter first string: ";
cin.getline(input1, 100);
cout << "Enter second string: ";
cin.getline(input2, 100);
MyString s(input1, input2);
s.display();
```

return 0;

}

Output:

```
Enter first string: World
Enter second string: Cup
Dynamic constructor called. Strings joined.
Joined String: WorldCup
Destructor called. Memory released.
```

10. Write a program to create an array of five Employee objects, each with name and salary. Display the employee with the highest salary.

```
#include <iostream>

using namespace std;

class Employee {

private:

string name;

float salary;

public:

void input() {

cout << "Enter employee name: ";

cin.ignore(); // clear newline character

getline(cin, name);

cout << "Enter salary: ";

cin >> salary;

}

void display() const {

cout << "Name: " << name << ", Salary: Rs. " << salary << endl;

}

float getSalary() const {

return salary;

}

};
```

```
int main() {
Employee emp[5];
int highestIndex = 0;
for (int i = 0; i < 5; i++) {
cout << "\n--- Enter Details for Employee " << i + 1 << " ---" << endl;
emp[i].input();
if (emp[i].getSalary() > emp[highestIndex].getSalary()) {
highestIndex = i;
}
}
cout << "\n--- Employee with Highest Salary ---" << endl;
emp[highestIndex].display();
return 0;
}
```

Output:

```
--- Enter Details for Employee 1 ---
Enter employee name: Sachin Jha
Enter salary: 35000

--- Enter Details for Employee 2 ---
Enter employee name: Sarthak Bhattarai
Enter salary: 39000

--- Enter Details for Employee 3 ---
Enter employee name: Mukesh Pandeya
Enter salary: 23000

--- Enter Details for Employee 4 ---
Enter employee name: Nawnit Poudel
Enter salary: 24000

--- Enter Details for Employee 5 ---
Enter employee name: Rohit Sharma
Enter salary: 45000

--- Employee with Highest Salary ---
Name: Rohit Sharma, Salary: Rs. 45000
```

11. Write a program to define a class Point. Use pointers to dynamically allocate memory for two points and calculate the distance between them.

```
#include <iostream>
#include <cmath>
using namespace std;
class Point {
private:
float x, y;
public:
Point(float a = 0, float b = 0) {
x = a;
y = b;
}
void input() {
cout << "Enter x and y coordinates: ";
cin >> x >> y;
}
friend float distance(Point* p1, Point* p2);
};
float distance(Point* p1, Point* p2) {
return sqrt(pow(p2->x - p1->x, 2) + pow(p2->y - p1->y, 2));
}
int main() {
Point* point1 = new Point();
Point* point2 = new Point();
cout << "--- Enter coordinates for Point 1 ---" << endl;
point1->input();
cout << "\n--- Enter coordinates for Point 2 ---" << endl;
point2->input();
float d = distance(point1, point2);
cout << "\nDistance between the two points: " << d << endl;
delete point1;
```

```
delete point2;

return 0;

}
```

Output:

```
--- Enter coordinates for Point 1 ---
Enter x and y coordinates: 3
6

--- Enter coordinates for Point 2 ---
Enter x and y coordinates: 5
3

Distance between the two points: 3.60555
```

12. Write a program to define a class Box. Use the this pointer in a member function to compare two boxes and return the one with the greater volume.

```
#include <iostream>

using namespace std;

class Box {

private:

float length, width, height;

public:

Box(float l = 1, float w = 1, float h = 1) {

length = l;

width = w;

height = h;

}

void input() {

cout << "Enter length, width, and height: ";

cin >> length >> width >> height;

}

float volume() const {

return length * width * height;

}

Box compare(const Box& b) const {
```

```
if (this->volume() > b.volume()) {
return *this;
} else {
return b;
}
}
void display() const {
cout << "Dimensions (LxWxH): " << length << " x " << width << " x " << height << endl;
cout << "Volume: " << volume() << endl;
}
};
int main() {
Box box1, box2;
cout << "--- Enter details for Box 1 ---" << endl;
box1.input();
cout << "\n--- Enter details for Box 2 ---" << endl;
box2.input();
Box bigger = box1.compare(box2); // compare using 'this' pointer
cout << "\n--- Box with Greater Volume ---" << endl;
bigger.display();
return 0;
}
```

Output:

```
--- Enter details for Box 1 ---
Enter length, width, and height: 3
5
8

--- Enter details for Box 2 ---
Enter length, width, and height: 4
6
3

--- Box with Greater Volume ---
Dimensions (LxWxH): 3 x 5 x 8
Volume: 120
```

Discussions:

While performing the lab task, we found out the actual uses of different constructors, arrays and pointer of objects, dynamic memory allocation, static and constant members, friend classes and functions. And also, we realized the importance of correct syntax while writing the program code. With the use of these syntaxes, we found our program to be more usable, modular and efficient.

Conclusion:

Hence, we successfully understood and implemented the advanced object-oriented programming concepts in C++, including object manipulation through functions, arrays and pointers of objects, dynamic memory allocation, constructors, static and constant members, and friend functions and classes.