Write a program that creates a class called arc. The data members of the class are radius, angle and arc_length of the arc is provided. Write three functions that set the respective values, and then write a single constant function that will read the values.

```
#include<iostream>
#include<string>
using namespace std;
class arc
{
private:
	double radius;
	double angle;
	double arc_length;
public:
	void  setradius(double r)
	{
		radius = r;
	}
	void setangle(double a)
	{
		angle = a;
	}
	void setarc_length(double length)
	{
		arc_length = length;
	}
	void displayarcinfo() const
	{
		cout << "Radius:" << radius << endl;
		cout << "Angle:" << angle << endl;
		cout << "arclength:" << arc_length << endl;
	}
};
int main()
{
	arc myarc;
	myarc.setradius(6.7890);
	myarc.setangle(5.887);
	myarc.setarc_length(1.4578);
	myarc.displayarcinfo();
	return 0;
}
```

```
Radius:6.789
Angle:5.887
arclength:1.4578

C:\Users\HP\source\repos\Project130
To automatically close the console
```

Create a class Android_Device. The data members of the class are IMEIno (int), Type (String), Make (String), Modelno (int), Memory(float), Operating_System(String). Then Implement member functions to:

1. **Set the values of all data members.**

2. **Display the values of all data members**

```
#include<iostream>
#include<string>
using namespace std;
class Android_device
{
private:
	int IMEIno;
	string type;
	int model_no;
	float memory;
	string operating_system;
public:
	void  setIMEIno(int imei)
	{
		IMEIno = imei;
	}
	void settype(string typ)
	{
		type = typ;
	}
	void setmodel_no(int model)
	{
		model_no = model;
	}
	void setmemory(float memo)
	{
		memory = memo;
	}
	void setoperating_system(string operating)
	{
		operating_system = operating;
	}
	void displaydeviceinfo() const
	{
		cout << "Android device details:" << endl;
		cout << "imei:" << IMEIno << endl;
		cout << "type:" << type << endl;
		cout << "modelno:" << model_no << endl;
		cout << "memory:" << memory << endl;
		cout << "operating system:" << operating_system << endl;
	}
};
int main()
{
	Android_device android;
	android.setIMEIno(6.7890);
	android.settype("smartphone");
	android.setmodel_no(2023);
	android.setmemory(64.7);
	android.setoperating_system("android 12");
	android.displaydeviceinfo();
	return 0;
}
```

```
Android device details:
imei:6
type:smartphone
modelno:2023
memory:64.7
operating system:android 12

C:\Users\HP\source\repos\Project130\x64\De
To automatically close the console when de
```

OR

```cpp
#include<iostream>
#include<string>
using namespace std;
class Android_Device {
private:
    int IMEIno;
    std::string Type;
    std::string Make;
    int Modelno;
    float Memory;
    std::string Operating_System;

public:
    void setDeviceDetails(int imei, string type, int model, float memory, string os)
    {
        IMEIno = imei;
        Type = type;
        Modelno = model;
        Memory = memory;
        Operating_System = os;
    }
    void displayDeviceDetails()
    {
        cout << "Device Details:" << endl;
        cout << "IMEI Number: " << IMEIno << endl;
        cout << "Type: " << Type << endl;
        cout << "Model Number: " << Modelno << endl;
        cout << "Memory: " << Memory << " GB" << endl;
        cout << "Operating System: " << Operating_System << endl;
    }
};

int main() {
    Android_Device myDevice;
    myDevice.setDeviceDetails(12, "Smartphone", 2023, 64.0, "Android 12");
    myDevice.displayDeviceDetails();

    return 0;
   }
```

```
Device Details:
IMEI Number: 12
Type: Smartphone
Model Number: 2023
Memory: 64 GB
Operating System: Android 12
```

Write a class called quadrilateral. Your task is to store the length of all the sides of the quadrilateral and the value of the 2 opposite angles within the quadrilateral. Then implement member functions:

1. **To compute the area of the quadrilateral.**
2. **To compute the parameter of the quadrilateral.**
3. **A constant function that will display the length of all the sides, the angles, the parameter of the quadrilateral and area of the quadrilateral.**
4. **Create setter and getter methods.**

Demonstrate the use of the object in the main function. Make sure that the function names are meaningful and self-descriptive.

```
#include<iostream>
#include<string>
#include<cmath>
using namespace std;
class Quadrilateral
{
private:
	double side1, side2, side3, side4;
	double angle1, angle2;
public:
	void setsides(double s1, double s2, double s3, double s4)
	{
		side1 = s1;
		side2 = s2;
		side3 = s3;
		side4 = s4;
	}
	void setangles(double a1, double a2)
	{
			angle1 = a1;
			angle2 = a2;
	}
	double computearea() const
	{
		return 0.5 * side1 * side2 * sin(angle1);
	}
	double  computePerimeter() const
	{
		return side1 + side2 + side3 + side4;
	}
	void displayDetails() const {
		cout << "Sides: " << side1 << ", " << side2 << ", " << side3 << ", " <<
side4 << endl;
		cout << "Angles: " << angle1 << " degrees, " << angle2 << " degrees" <<
endl;
		cout << "Area: " << computearea() << endl;
		cout << "Perimeter: " << computePerimeter() << endl;
	}
};
int main() {
	Quadrilateral myQuadrilateral;
	myQuadrilateral.setsides(4.0, 5.0, 6.0, 7.0);
```

```
        myQuadrilateral.setangles(60.0, 120.0);

        myQuadrilateral.displayDetails();

        return 0;
    }
```

```
Sides: 4, 5, 6, 7
Angles: 60 degrees, 120 degrees
Area: -3.04811
Perimeter: 22

C:\Users\HP\source\repos\Project130\x64\Debug\P
To automatically close the console when debuggi
le when debugging stops.
```

Create a class "Vehicle" having:

- **"LisencePlate_No", "Model_No", "Type", and "Color" as private data members.**
- **Parameterized constructor assigning default values to the data members**
- **No-Argument constructor assigning default values to the data members**
- **Default constructor assigning default values to the data members**

```
#include<iostream>
#include<string>
using namespace std;
class Vehicle
{
private:
    string LicensePlate_No;
    string Model_No;
    string Type;
    string Color;
public:
    //parameterized constructor
    Vehicle(string licenseplate, string model, string type, string colour)
    {
        LicensePlate_No = licenseplate;
        Model_No = model;
        Type = type;
        Color = colour;
    }
    //No argument constructor
    Vehicle() :LicensePlate_No(), Model_No(), Type(), Color()
    {
        cout << "No argument constructor:" << endl;
    }
    void DisplayDetails()
    {
        cout << "Vehicle Details:" << endl;
        cout << "License Plate No: " << LicensePlate_No << endl;
        cout << "Model No: " << Model_No << endl;
        cout << "Type: " << Type << endl;
        cout << "Color: " << Color << endl;
    }
};
```

```
int main() {
    Vehicle myVehicle1; // Using the no-argument constructor
    myVehicle1.DisplayDetails();

    Vehicle myVehicle2("ABC123", "Sedan", "Honda", "Blue"); // Using the
parameterized constructor
    myVehicle2.DisplayDetails();

    return 0;
}
```

```
No argument constructor:
Vehicle Details:
License Plate No:
Model No:
Type:
Color:
Vehicle Details:
License Plate No: ABC123
Model No: Sedan
Type: Honda
Color: Blue
```

Write a C++ program for a new ice cream vendor called PopSalon. The management of PopSalon has decided that they are going to sell their popcorn in 11 different flavors namely: chocolate, English toffee, salted caramel, caramel, jalapeno, cheese, spiced cheese, plain sated, buttered, salt and pepper, and garlic. Carefully design the program by observing the following rules.

- **PopSalon is charging Rs. 100 for small pack, Rs. 250 for medium sized pack, Rs. 500 for large sized pack and Rs. 750 large size tin pack. Hence you will need a function to determine the size of the pack and based on that the price. If a user enters option number other than the ones displayed, your program should display an invalid input message and ask the user to re-enter the option number.**
- **PopSalon allows its customers to purchase a gift wrapper with their popcorn. If the customer wants to purchase the wrapper he will have to pay an additional Rs 50. This amount should be added to the total amount payable by the user.**
- **If the customer asks for chocolate sauce, caramel sauce, or melted cheese as an additional topping then he will have to pay an additional amount of Rs. 50, Rs. 30 and, Rs. 60 respectively. Design a function that will be called if the customer chooses an additional topping.**
- **The program should show a menu that asks the customer for his requirements and then displays the final payable amount with full details about the flavor, the size of the pack and details regarding any additional toppings.**
- **For service quality inspection and monthly product promotions, the program should ask the user to enter his/her contact details including name, mobile number and email address, and select between the options good, neutral and bad against the quality of the service provided.**
- **In the end create a class destructor that displays a thank you message to the user. Design your program using sound OOP practices. Carefully determine the data members, member functions, access specifiers, activities to be performed in the constructor. Make sure that you use good naming conventions in your code. A good design can earn you higher marks.**

1.1 Outcome:

After completing this lab, students will be able to design a class that correctly implements class members by observing access specifiers. The students will also be familiar with the use of a constructor and destructor.

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
class PopcornOrder
{
private:
    string flavor;
    int packSize;
    int totalAmount;
    bool giftWrapper;
    vector<string> additionalToppings;
    string contactName;
    string contactNumber;
    string email;
    string serviceQuality;

public:
    PopcornOrder()
    {
        totalAmount = 0;
        packSize = 0;
        giftWrapper = false;
    }
    void selectFlavor()
    {
        cout << "Select Popcorn Flavor:" << endl;
        vector<string> flavors =
        {
            "Chocolate", "English Toffee", "Salted Caramel", "Caramel", "Jalapeno",
            "Cheese", "Spiced Cheese", "Plain Salted", "Buttered", "Salt and
Pepper", "Garlic"
        };

        for (int i = 0; i < flavors.size(); ++i)
        {
            cout << i + 1 << ". " << flavors[i] << endl;
        }

        int choice;
        cout << "Enter the flavor number: ";
        cin >> choice;

        if (choice < 1 || choice > flavors.size())
        {
            cout << "Invalid input. Please re-enter the option number." << endl;
            selectFlavor();
        }
        else
        {
```

```
            flavor = flavors[choice - 1];
        }
    }
    void selectPackSize()
    {
        cout << "Select Popcorn Pack Size:" << endl;
        cout << "1. Small Pack (Rs. 100)\n2. Medium Pack (Rs. 250)\n3. Large Pack
(Rs. 500)\n4. Large Tin Pack (Rs. 750)" << endl;
        int choice;
        cout << "Enter the pack size number: ";
        cin >> choice;

        switch (choice) {
        case 1:
            packSize = 100;
            break;
        case 2:
            packSize = 250;
            break;
        case 3:
            packSize = 500;
            break;
        case 4:
            packSize = 750;
            break;
        default:
            cout << "Invalid input. Please re-enter the option number." << endl;
            selectPackSize();
        }
    }
    void addToppings()
    {
        cout << "Do you want additional toppings?" << endl;
        vector<string> toppings =
        {
            "Chocolate Sauce (Rs. 50)", "Caramel Sauce (Rs. 30)", "Melted Cheese
(Rs. 60)"
        };

        for (int i = 0; i < toppings.size(); ++i)
        {
            cout << i + 1 << ". " << toppings[i] << endl;
        }
        int choice;
        cout << "Enter the topping number (0 to finish adding toppings): ";
        cin >> choice;
        if (choice < 0 || choice > toppings.size()) {
            cout << "Invalid input. Please re-enter the option number." << endl;
            addToppings();
        }
        else if (choice > 0)
        {
            additionalToppings.push_back(toppings[choice - 1]);
            totalAmount += getPriceForTopping(toppings[choice - 1]);
            addToppings();
        }
    }
    int getPriceForTopping(const string& topping)
```

```
{
    if (topping.find("Chocolate Sauce") != string::npos)
    {
        return 50;
    }
    else if (topping.find("Caramel Sauce") != string::npos)
    {
        return 30;
    }
    else if (topping.find("Melted Cheese") != string::npos)
    {
        return 60;
    }
    return 0;
}
void purchaseGiftWrapper()
{
    char choice;
    cout << "Do you want to purchase a gift wrapper? (Y/N): ";
    cin >> choice;
    if (choice == 'Y' || choice == 'y')
    {
        giftWrapper = true;
        totalAmount += 50;
    }
}
void collectContactDetails()
{
    cout << "Enter your contact details:" << endl;
    cout << "Name: ";
    cin.ignore();
    getline(cin, contactName);
    cout << "Mobile Number: ";
    cin >> contactNumber;
    cout << "Email Address: ";
    cin >> email;
    cout << "Rate the service quality:" << endl;
    cout << "1. Good\n2. Neutral\n3. Bad" << endl;
    int choice;
    cout << "Enter the rating number: ";
    cin >> choice;

    switch (choice)
    {
    case 1:
        serviceQuality = "Good";
        break;
    case 2:
        serviceQuality = "Neutral";
        break;
    case 3:
        serviceQuality = "Bad";
        break;
    default:
        serviceQuality = "Invalid";
    }
}
void displayOrderDetails()
```

```
    {
        cout << "PopSalon - Order Details:" << endl;
        cout << "Flavor: " << flavor << endl;
        cout << "Pack Size: " << packSize << " (Rs. " << packSize << ")" << endl;
        cout << "Toppings: ";
        if (additionalToppings.empty())
        {
            cout << "None";
        }
        else
        {
            for (const string& topping : additionalToppings)
            {
                cout << topping << ", ";
            }
        }
        cout << endl;
        if (giftWrapper)
        {
            cout << "Gift Wrapper: Yes (Rs. 50)" << endl;
        }
        else
        {
            cout << "Gift Wrapper: No" << endl;
        }

        cout << "Total Amount Payable: Rs. " << totalAmount << endl;

        cout << "Contact Details:" << endl;
        cout << "Name: " << contactName << endl;
        cout << "Mobile Number: " << contactNumber << endl;
        cout << "Email Address: " << email << endl;
        cout << "Service Quality: " << serviceQuality << endl;
    }
};

int main() {
    PopcornOrder order;
    order.selectFlavor();
    order.selectPackSize();
    order.addToppings();
    order.purchaseGiftWrapper();
    order.collectContactDetails();
    order.displayOrderDetails();

    cout << "Thank you for visiting PopSalon!" << endl;
    return 0;
}
```

Your task is to create a class called examination. The class has data member's duration, credit_hours, course title, month, date, year and time. Your task is to create the individual member functions and call them using the class constructor. Be very vigilant in determining the access specifiers for the data members and member functions.

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

```cpp
class Examination
{
private:
    int duration;
    int creditHours;
    string courseTitle;
    int month;
    int date;
    int year;
    string time;

public:
    Examination(int dur, int credits, string title, int m, int d, int y, string t)
    {
        duration = dur;
        creditHours = credits;
        courseTitle = title;
        month = m;
        date = d;
        year = y;
        time = t;
    }
    void displayExaminationDetails()
    {
        cout << "Exam Details:" << endl;
        cout << "Duration: " << duration << " minutes" << endl;
        cout << "Credit Hours: " << creditHours << endl;
        cout << "Course Title: " << courseTitle << endl;
        cout << "Date: " << month << "/" << date << "/" << year << endl;
        cout << "Time: " << time << endl;
    }
};

int main()
{
    Examination exam(120, 3, "Mathematics", 10, 22, 2023, "10:00 AM");
    exam.displayExaminationDetails();

    return 0;
}
```

```
Exam Details:
Duration: 120 minutes
Credit Hours: 3
Course Title: Mathematics
Date: 10/22/2023
Time: 10:00 AM
```

Write a program that creates a class called student. The data members of the class are name and age.

- **Create a nullary constructor and initialize the class object.**
- **Create a parameterized constructor that can set the values being passed from the main function.**
- **Create a display function called showAll( ) which will be used to show values that have been set.**

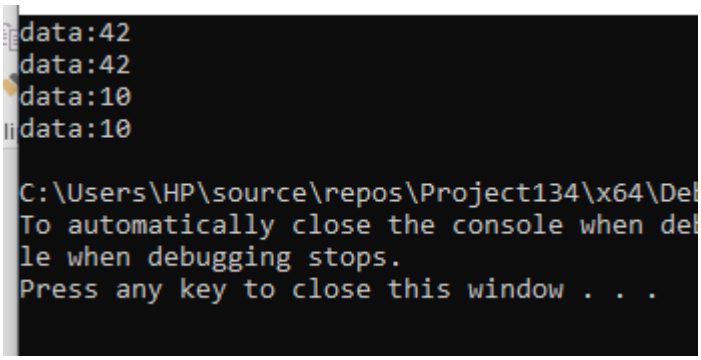
Use the default copy constructor to show that copying of simple objects can be accomplished through the use of the default copy constructor.

```cpp
#include <iostream>
#include <string>
using namespace std;
class Student
{
private:
    string name;
    int age;

public:
    // Nullary constructor
    Student()
    {
        name = "Unknown";
        age = 0;
    }

    // Parameterized constructor
    Student(string studentName, int studentAge) {
        name = studentName;
        age = studentAge;
    }

    // Display function to show student details
    void showAll()
    {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << " years" << endl;
    }
};

int main() {
    // Create a student object using the nullary constructor
    Student student1;

    // Create another student object using the parameterized constructor
    Student student2("John Doe", 20);

    // Display details of student1
    cout << "Student 1 Details:" << endl;
    student1.showAll();

    // Display details of student2
    cout << "Student 2 Details:" << endl;
    student2.showAll();

    // Use the default copy constructor to create a copy of student2
    Student student3 = student2;

    // Display details of student3 (copy of student2)
    cout << "Student 3 Details (Copy of Student 2):" << endl;
    student3.showAll();

    return 0;
}
```

```
Name: Unknown
Age: 0 years
Student 2 Details:
Name: John Doe
Age: 20 years
Student 3 Details (Copy of Student 2):
Name: John Doe
Age: 20 years
```

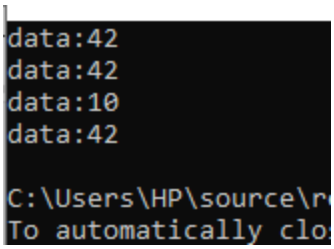
SHALLOW COPY CONSTRUCTOR

```
#include<iostream>
using namespace std;
class shallowcopyexample
{
private:
	int* data;
public:
	shallowcopyexample(int value)
	{
		data = new int(value);
	}
	shallowcopyexample(const shallowcopyexample& other)
	{
		data = other.data;
	}
	void setdata(int value)
	{
		*data = value;
	}
	void displaydata()
	{
		cout << "data:" << *data << endl;
	}
	~shallowcopyexample()
	{
		delete data;
	}
};
int main()
{
	shallowcopyexample original(42);
	shallowcopyexample copy(original);
	original.displaydata();
	copy.displaydata();
	original.setdata(10);
	original.displaydata();
	copy.displaydata();
	return 0;
}
```



DEEP COPY CONSTRUCTOR:

```
#include<iostream>
using namespace std;
class deepcopyexample
{
private:
	int* data;
public:
	deepcopyexample(int value)
	{
		data = new int(value);
	}
	deepcopyexample(const deepcopyexample& other)
	{
		data = new int (*(other.data));
	}
	void setdata(int value)
	{
		*data = value;
	}
	void displaydata()
	{
		cout << "data:" << *data << endl;
	}
	~deepcopyexample()
	{
		delete data;
	}
};
int main()
{
	deepcopyexample original(42);
	deepcopyexample copy(original);
	original.displaydata();
	copy.displaydata();
	original.setdata(10);
	original.displaydata();
	copy.displaydata();
	return 0;
}
```



WORKING WITH ARRAY(JUST FOR UNDERSTANDING):

```
#include<iostream>
using namespace std;
int main()
{
	float arr[3];
	float* ptr;
	cout << "Displaying address using array:" << endl;
	for (int i = 0;i < 3;i++)
	{
		cout << "&arr[" << i << "] = " << &arr[i] << endl;
	}
	ptr = arr;
	cout << "Displaying address using pointer:" << endl;
	for (int i = 0;i < 3;i++)
	{
		cout << "ptr+" << i << "=" << ptr + i << endl;
	}
}
```

LAB:6
PRACTICE TASK:2

```
#include<iostream>
using namespace std;
class polygon
{
private:
	float length;
	float* width;
public:
	polygon(float len, float wid)
	{
		length = len;
		width = new float(wid);
	}
	polygon(const polygon& other)
	{
		length = other.length;
		width = other.width;
	}
	void setwidth(float wid)
	{
		*width = wid;
	}
	void displayall()
	{
```

```
		cout << "length:" << length << " " << "width:" << *width << endl;
	}
	~polygon()
	{
		delete width;
	}
};
int main()
{
	polygon one(1.5, 4.2);
	one.displayall();
	polygon two(one);
	two.displayall();
	one.setwidth(1.3);
	one.displayall();
	two.displayall();
	return 0;
}
```

```
length:1.5 width:4.2
length:1.5 width:4.2
length:1.5 width:1.3
length:1.5 width:1.3

C:\Users\HP\source\repos\Project1
To automatically close the consol
le when debugging stops.
```

FILE HANDLING:

CREATES AND WRITES TO A FILE:

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
	string data;
	ofstream myfile;
	myfile.open("filename.txt");
	cout << "Enter student name:" << endl;
	cin >> data;
	myfile << data << endl;
	cout << "Enter student age:" << endl;
	cin >> data;
	myfile << data << endl;
	cout << "Enter program:" << endl;
	cin >> data;
	myfile << data << endl;
	myfile.close();
	return 0;
}
```

READS A FILE: