Salman Ahmad

04072113050

BSCS 6th Sem

CS-121 OOP Assignment 1

Q1.

```
#include<iostream>
#include<string>
using namespace std;
class Android_Device{

	int IMEIno;
	string Type;
	string Make;
	int Modelno;
	float Memory;
	string Operating_System;

	public:
		//setters
	void setIMEIno(int IMI){
		IMEIno = IMI;
	}
	void setType(string Type){
		this->Type=Type;
```

```
}
void setMake(string Make){
        this->Make=Make;
}
void setMemory(float Memory){
        this->Memory = Memory;
}
void setOS(string OS){
        Operating_System=OS;
}



//getter
int getIMEIno(){
        return IMEIno;
}
string getType(){
        return Type;
}
string getMake(){
        return Make;
}
float getMemory(){
        return Memory;
}
string getOS(){
        return Operating_System;
}
```

```
};
int main(){
	Android_Device s1;
	s1.setIMEIno(040);
	s1.setMake("Samsung");
	s1.setType("Solo");
	s1.setMemory(128);
	s1.setOS("Linux");
	cout<<"IMEI No is:"<<s1.getIMEIno()<<endl;
	cout<<"Make of the mobile is:"<<s1.getMake()<<endl;
	cout<<"Type of the mobile is:"<<s1.getType()<<endl;
	cout<<"Memory of the mobile is:"<<s1.getMemory()<<endl;
	cout<<"Operating system of the device is:"<<s1.getOS()<<endl;





	return 0;
}
```

```
C:\Users\salma\Desktop\OOP

IMEI No is:32
Make of the mobile is:Samsung
Type of the mobile is:Solo
Memory of the mobile is:128
Operating system of the device is:Linux

--------------------------------
Process exited after 0.04312 seconds with return value 0
Press any key to continue . . .
```

Q2.

```
#include <iostream>

#include <cmath>  // For sqrt

#include <iomanip>


using namespace std;


class Quadrilateral {

private:

   double side1, side2, side3, side4;

   double angle1, angle2;


public:

   // Default Constructor

   Quadrilateral() : side1(0), side2(0), side3(0), side4(0), angle1(0), angle2(0) {}


   // Parameterized Constructor

   Quadrilateral(double s1, double s2, double s3, double s4, double a1, double a2)
```

```
    : side1(s1), side2(s2), side3(s3), side4(s4), angle1(a1), angle2(a2) {}

// Setter methods
void setSide1(double s1) {
    side1 = s1;
}
void setSide2(double s2) {
    side2 = s2;
}
void setSide3(double s3) {
    side3 = s3;
}
void setSide4(double s4) {
    side4 = s4;
}
void setAngle1(double a1) {
    angle1 = a1;
}
void setAngle2(double a2) {
    angle2 = a2;
}



// Getter methods
double getSide1() const {
    return side1;
}
double getSide2() const {
```

```
    return side2;
}
double getSide3() const {
    return side3;
}
double getSide4() const {
    return side4;
}

double getAngle1() {
    return angle1;
}
double getAngle2() {
    double angle2;
}


//Perimeter being computed here
double calculatePerimeter() {
    return side1 + side2 + side3 + side4;
}

// Method to compute the perimeter
double calculatePerimeter() const {
    return side1 + side2 + side3 + side4;
}

// Method to compute the area
double calculateArea() const {
```

```
    double s = calculatePerimeter() / 2;
    double area = sqrt((s - side1) * (s - side2) * (s - side3) * (s - side4) -
        0.5 * side1 * side2 * side3 * side4 *
        (cos(angle1 * 3.14 / 180.0) * cos(angle2 * 3.14 / 180.0)));
    return area;
  }

  // Method to display all properties
  void display() const {
    cout << "Sides of the Quadrilateral: \n";
    cout << "Side 1: " << side1 << "\n";
    cout << "Side 2: " << side2 << "\n";
    cout << "Side 3: " << side3 << "\n";
    cout << "Side 4: " << side4 << "\n";
    cout << "Angles of the Quadrilateral: \n";
    cout << "Angle 1: " << angle1 << " degrees\n";
    cout << "Angle 2: " << angle2 << " degrees\n";
    cout << "Perimeter: " << calculatePerimeter() << "\n";
    cout << "Area: " << fixed << calculateArea() << "\n";
  }
};

int main() {
  Quadrilateral quad;

  // Initialization
  quad.setSide1(30);
  quad.setSide2(150);
  quad.setSide3(140);
```

```
    quad.setSide4(20);

    quad.setAngle1(80);
    quad.setAngle2(110);

    // Display the properties of the quadrilateral
    quad.display();

    return 0;
}
```

```
Sides of the Quadrilateral:
Side 1: 30
Side 2: 150
Side 3: 140
Side 4: 20
Angles of the Quadrilateral:
Angle 1: 80 degrees
Angle 2: 110 degrees
Perimeter: 340
Area: 3602.035759

--------------------------------
Process exited after 0.0397 seconds with return value 0
Press any key to continue . . .
```

Q3.

```
#include <iostream>
#include <string>

using namespace std;

class Vehicle {
private:
```

```
    string licencePlateNo;
    string modelNo;
    string type;
    string color;

public:
    // Parameterized constructor
    Vehicle(const string& plate, const string& model, const string& type, const string& color)
        : licencePlateNo(plate), modelNo(model), type(type), color(color) {}

    // Default constructor
    Vehicle() : licencePlateNo("NULL"), modelNo("NULL"), type("NULL"), color("NULL") {}

    // Setter methods
    void setLicencePlateNo(const string& plate) {
        licencePlateNo = plate;
    }

    void setModelNo(const string& model) {
        modelNo = model;
    }

    void setType(const string& type) {
        this->type = type;
    }

    void setColor(const string& color) {
        this->color = color;
    }
```

```
// Getter methods
string getLicencePlateNo() const {
   return licencePlateNo;
}

string getModelNo() const {
   return modelNo;
}

string getType() const {
   return type;
}

string getColor() const {
   return color;
}

// Public member functions
void RegisterVehicle() const {
   cout << "Vehicle registered successfully." << endl;
}

void UpdateVehicle() const {
   cout << "Vehicle details updated successfully." << endl;
}

void DeleteVehicle() const {
   cout << "Vehicle deleted successfully." << endl;
```

```
    }

    void ShowVehicleInfo() const {
        cout << "Vehicle Info: " << endl;
        cout << "Licence Plate No: " << licencePlateNo << endl;
        cout << "Model No: " << modelNo << endl;
        cout << "Type: " << type << endl;
        cout << "Color: " << color << endl;
    }
};

// Function to find a vehicle by licence plate number
int findVehicleIndex(const Vehicle vehicles[], int size, const string& plate) {
    for (int i = 0; i < size; ++i) {
        if (vehicles[i].getLicencePlateNo() == plate) {
            return i;
        }
    }
    return -1; // Vehicle not found
}

int main() {
    const int MAX_VEHICLES = 100;
    Vehicle vehicles[MAX_VEHICLES];
    int vehicleCount = 0;
    bool running = true;
    int choice;
    string plate, model, type, color;
```

```
while (running) {
    cout << "\nVehicle Management System\n";
    cout << "1. Register Vehicle\n";
    cout << "2. Update Vehicle\n";
    cout << "3. Delete Vehicle\n";
    cout << "4. Search Vehicle or Show All Vehicles\n";
    cout << "5. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            if (vehicleCount < MAX_VEHICLES) {
                cout << "Enter Licence Plate No: ";
                cin >> plate;
                cout << "Enter Model No: ";
                cin >> model;
                cout << "Enter Type: ";
                cin >> type;
                cout << "Enter Color: ";
                cin >> color;
                vehicles[vehicleCount++] = Vehicle(plate, model, type, color);
                vehicles[vehicleCount - 1].RegisterVehicle();
            } else {
                cout << "Vehicle storage is full." << endl;
            }
            break;

        case 2:
```

```
    cout << "Enter Licence Plate No of the vehicle to update: ";
    cin >> plate;
    {
        int index = findVehicleIndex(vehicles, vehicleCount, plate);
        if (index != -1) {
            cout << "Enter new Model No: ";
            cin >> model;
            vehicles[index].setModelNo(model);
            cout << "Enter new Type: ";
            cin >> type;
            vehicles[index].setType(type);
            cout << "Enter new Color: ";
            cin >> color;
            vehicles[index].setColor(color);
            vehicles[index].UpdateVehicle();
        } else {
            cout << "Vehicle not found." << endl;
        }
    }
    break;

case 3:
    cout << "Enter Licence Plate No of the vehicle to delete: ";
    cin >> plate;
    {
        int index = findVehicleIndex(vehicles, vehicleCount, plate);
        if (index != -1) {
            vehicles[index].DeleteVehicle();
            // Shift vehicles to remove the deleted vehicle
```

```
        for (int i = index; i < vehicleCount - 1; ++i) {
            vehicles[i] = vehicles[i + 1];
        }
        --vehicleCount;
    } else {
        cout << "Vehicle not found." << endl;
    }
}
break;

case 4:
    cout << "Enter Licence Plate No of the vehicle to search (or type 'all' to show all vehicles): ";
    cin >> plate;
    if (plate == "all") {
        // Display all vehicles
        cout << "All Registered Vehicles:" << endl;
        for (int i = 0; i < vehicleCount; ++i) {
            vehicles[i].ShowVehicleInfo();
            cout << "-----------------------------" << endl;
        }
    } else {
        // Search for a specific vehicle
        int index = findVehicleIndex(vehicles, vehicleCount, plate);
        if (index != -1) {
            vehicles[index].ShowVehicleInfo();
        } else {
            cout << "Vehicle not found." << endl;
        }
    }
```

```
            break;

        case 5:
            running = false;
            cout << "Exiting the program." << endl;
            break;

        default:
            cout << "Invalid choice. Please enter a number between 1 and 5." << endl;
            break;
    }
}
 return 0;
```

}

```
Vehicle Management System
1. Register Vehicle
2. Update Vehicle
3. Delete Vehicle
4. Search Vehicle or Show All Vehicles
5. Exit
Enter your choice: 1
Enter Licence Plate No: 123
Enter Model No: 1
Enter Type: 1
Enter Color: red
Vehicle registered successfully.

Vehicle Management System
1. Register Vehicle
2. Update Vehicle
3. Delete Vehicle
4. Search Vehicle or Show All Vehicles
5. Exit
Enter your choice: 4
Enter Licence Plate No of the vehicle to search (or type 'all' to show all vehicles): 123
Vehicle Info:
Licence Plate No: 123
Model No: 1
Type: 1
Color: red

Vehicle Management System
1. Register Vehicle
```

```
Model No: 1
Type: 1
Color: red

Vehicle Management System
1. Register Vehicle
2. Update Vehicle
3. Delete Vehicle
4. Search Vehicle or Show All Vehicles
5. Exit
Enter your choice: 2
Enter Licence Plate No of the vehicle to update: 123
Enter new Model No: 10
Enter new Type: R
Enter new Color: Redd
Vehicle details updated successfully.

Vehicle Management System
1. Register Vehicle
2. Update Vehicle
3. Delete Vehicle
4. Search Vehicle or Show All Vehicles
5. Exit
Enter your choice: 4
Enter Licence Plate No of the vehicle to search (or type 'all' to show all vehicles): all
All Registered Vehicles:
Vehicle Info:
Licence Plate No: 123
Model No: 10
Type: R
```