Salman Ahmad

04072113050

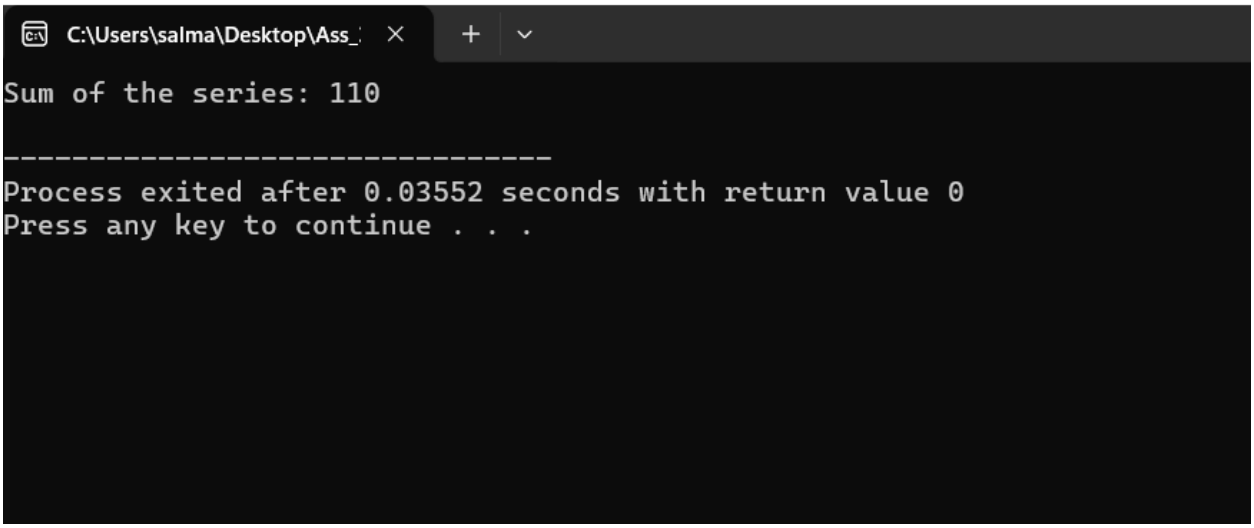
BSCS 6th Sem

CS-121 OOP Assignment 3

Q1.

```
using namespace std;

class ArithmeticSeries {
private:
   int firstEntry;
   int lastEntry;
   int n;    // Total number of terms
   int sum;  // Sum of the series

public:
   // Constructor to initialize the series
   ArithmeticSeries(int first, int last, int terms);

   friend void computeSum(ArithmeticSeries &series);

   void displaySum() const;
};

// Constructor implementation
```

```
ArithmeticSeries::ArithmeticSeries(int first, int last, int terms)
   : firstEntry(first), lastEntry(last), n(terms), sum(0) {}

// Friend function
void computeSum(ArithmeticSeries &series) {
   series.sum = (series.n / 2.0) * (series.firstEntry + series.lastEntry);
}

// Function to display the sum
void ArithmeticSeries::displaySum() const {
   cout << "Sum of the series: " << sum << endl;
}

int main() {
   //instance of ArithmeticSeries
   ArithmeticSeries series(2, 20, 10);
   computeSum(series);
   series.displaySum();

   return 0;}
```



Q2.

```
#include <iostream>
#include<string>
using namespace std;
class Rectangle {
private:
   double length;
   double width;
public:
   //mutator functions
   void setLength(double);
   void setWidth(double);
   // accessor functions
   double getLength() const;
   double getWidth() const;
   //Membor function recArea()
   double recArea() const;
};

class Cuboid : public Rectangle {
private:
   double height;

public:

   // Mutator
   void setHeight(double h);
   // Accessor method
```

```
    double getHeight() const;
    //volume
    double cuboidArea() const;

};


// class Rectangle methods
void Rectangle::setLength(double val)
{
    length = val;
}


void Rectangle::setWidth(double val)
{
    width = val;
}

double Rectangle::getLength() const
{
    return length;
}

double Rectangle::getWidth() const
{
    return length;
}
double Rectangle::recArea() const
```

```
{
   return            length * width;
}



//class Cuboid starts from here
void Cuboid::setHeight(double h) {
   height = h;
}
double Cuboid::getHeight() const {
   return height;
}

double Cuboid::cuboidArea() const {
   return recArea() * height; // Cuboid Area = length * width * height
}

int main() {
   Cuboid obj1;
   cout << "Setting the object length:10cm, object width:5cm, object height:20cm \n";
   obj1.setLength(10);
   obj1.setWidth(5);
   obj1.setHeight(20);
   cout << "\nNow getting Data members value through get methods:\n";
   cout << "Object height is:" << obj1.getHeight() << endl;
   cout << "Object length is:" << obj1.getLength() << endl;
   cout << "Object width is:" << obj1.getWidth() << endl;
   cout << "\n\nArea of Rectangle is:" << obj1.recArea() << endl;
   cout << "Area of Cuboid is:" << obj1.cuboidArea() << endl;
```

}

```
C:\Users\salma\Desktop\Ass_

Setting the object length:10cm, object width:5cm, object height:20cm

Now getting Data members value through get methods:
Object height is:20
Object length is:10
Object width is:10


Area of Rectangle is:50
Area of Cuboid is:1000

--------------------------------
Process exited after 0.03855 seconds with return value 0
Press any key to continue . . .
```

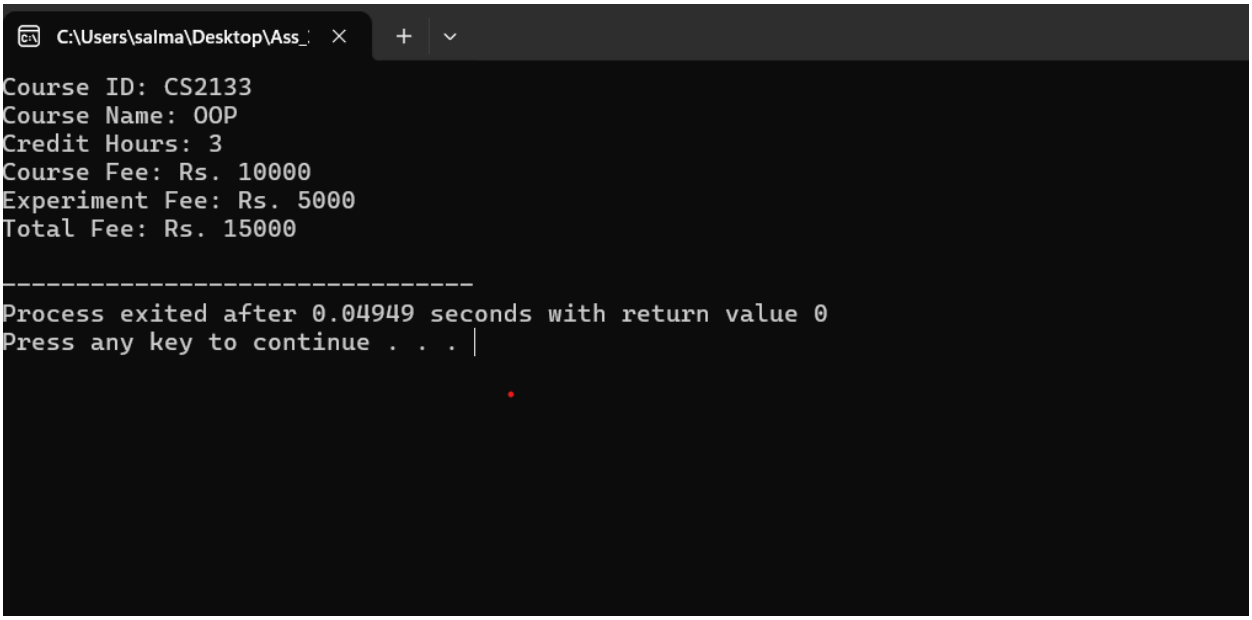
Q3.

```
#include <iostream>

#include <string>

using namespace std;


// Base class

class GraduateCourse {

protected:

   string courseID;

   string courseName;

   int creditHours;

   double courseFee;
```

```
public:
   // Parameterized constructor
   GraduateCourse(string id, string name, int hours, double fee);

   // Destructor
   ~GraduateCourse();

   // Getter functions
   string getCourseID() const;
   string getCourseName() const;
   int getCreditHours() const;
   double getCourseFee() const;
};

// Implementation of GraduateCourse functions
GraduateCourse::GraduateCourse(string id, string name, int hours, double fee)
   : courseID(id), courseName(name), creditHours(hours), courseFee(fee) {}

GraduateCourse::~GraduateCourse() {}

string GraduateCourse::getCourseID() const
{
   return courseID;
}
string GraduateCourse::getCourseName() const
{
   return courseName;
}
int GraduateCourse::getCreditHours() const
```

```
{
   return creditHours;
}
double GraduateCourse::getCourseFee() const
{
   return courseFee;
}

// Derived class
class ResearchCourse : public GraduateCourse {
private:
   double experimentFee;

public:
   // Parameterized constructor
   ResearchCourse(string id, string name, int hours, double fee, double expFee);

   // Functions
   void setExperimentFee(double expFee);
   void display() const;
   double totalFee() const;
};

// Implementation of ResearchCourse functions
ResearchCourse::ResearchCourse(string id, string name, int hours, double fee, double expFee)
   : GraduateCourse(id, name, hours, fee), experimentFee(expFee) {}

void ResearchCourse::setExperimentFee(double expFee) {
   experimentFee = expFee;
```

```
}

void ResearchCourse::display() const {
   cout << "Course ID: " << getCourseID() << endl;
   cout << "Course Name: " << getCourseName() << endl;
   cout << "Credit Hours: " << getCreditHours() << endl;
   cout << "Course Fee: Rs. " << getCourseFee() << endl;
   cout << "Experiment Fee: Rs. " << experimentFee << endl;
}

double ResearchCourse::totalFee() const {
   return getCourseFee() + experimentFee;
}

int main() {
   // Creating an instance of ResearchCourse
   ResearchCourse researchCourse("CS2133", "OOP", 3, 10000, 5000);

   // Displaying attributes and total fee
   researchCourse.display();
   cout << "Total Fee: Rs. " << researchCourse.totalFee() << endl;

   return 0;
}
```



Q4.

```
#include <iostream>

#include <string>

using namespace std;


// Base class

class Car {

protected:

   string carName;

   bool ignition;

   int currentSpeed;


public:

   // No-argument constructor

   Car();
```

```
    // Parameterized constructor
    Car(string name, bool ign, int speed);

    // Setter functions
    void setCarName(const string& name);
    void setIgnition(bool ign);
    void setCurrentSpeed(int speed);

    // Getter functions
    string getCarName() const;
    bool getIgnition() const;
    int getCurrentSpeed() const;

    // Function to set speed
    void setSpeed(int speed);
};

// Implementation of Car functions
Car::Car() :carName("Unknown"), ignition(false),currentSpeed(0) {}

Car::Car(string name, bool ign, int speed) : carName(name), ignition(ign), currentSpeed(speed) {}

void Car::setCarName(const string& name)
{
    carName = name;
}

void Car::setIgnition(bool ign)
```

```
{
   ignition = ign;
}

void Car::setCurrentSpeed(int speed)
{
   currentSpeed = speed;
}



//getters from here
string Car::getCarName() const
{
   return carName;
}

bool Car::getIgnition() const
{
   return ignition;
}

int Car::getCurrentSpeed() const
{
   return currentSpeed;
}

void Car::setSpeed(int speed)
{
   currentSpeed = speed;
```

```
}

// Derived class
class Convertible : public Car
{
private:
   bool top;

public:
   // No-argument constructor
   Convertible();

   // Four-argument constructor
   Convertible(string name, bool ign, int speed, bool t);

   // Setter for top
   void setTop(bool t);

   // Function to display all attributes
   void show() const;
};

// Implementation of Convertible functions
Convertible::Convertible() : Car(), top(false) {}

Convertible::Convertible(string name, bool ign, int speed, bool t) : Car(name, ign, speed), top(t) {}

void Convertible::setTop(bool t) {
   top = t;
```

```
}

void Convertible::show() const {
    cout << "Car Name: " << getCarName() << endl;
    cout << "Ignition: " << (getIgnition() ? "On" : "Off") << endl;
    cout << "Current Speed: " << getCurrentSpeed() << " km/h" << endl;
    cout << "Top: " << (top ? "Down" : "Up") << endl;
}

int main() {
    // Creating an instance of Convertible
    Convertible myConvertible("Ford Mustang", true, 120, true);

    // Displaying attributes
    myConvertible.show();



    cout << "\n\nNow updating values:\n";
    // Testing setter functions
    myConvertible.setCarName("Fortuner");
    myConvertible.setIgnition(false);
    myConvertible.setCurrentSpeed(80);
    myConvertible.setTop(false);

    // Displaying
    myConvertible.show();

    return 0;
}
```

C:\Users\salma\Desktop\Ass_

```
Car Name: Ford Mustang
Ignition: On
Current Speed: 120 km/h
Top: Down


Now updating values:
Car Name: Fortuner
Ignition: Off
Current Speed: 80 km/h
Top: Up

--------------------------------
Process exited after 0.04526 seconds with return value 0
Press any key to continue . . .
```