



NAME: M AUN RABBANI

REG NO: BCS231032

ASSIGNMENT NO: 03

SUBMITTED TO:

SIR ADNAN JELANI

PROGRAM 1:

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <string>
```

```
using namespace std;
```

```
class Student {
```

```
private:
```

```
    string name;
```

```
    int roll_number;
```

```
    float marks;
```

```
public:
```

```
    void input() {
```

```
        cout << "Enter name: ";
```

```
        cin >> name;
```

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

```
        cin >> roll_number;
```

```
        cout << "Enter marks: ";
```

```
        cin >> marks;
```

```
    }
```

```
    void display() const {
```

```
        cout << "Name: " << name << endl;
```

```
        cout << "Roll Number: " << roll_number << endl;
```

```
        cout << "Marks: " << marks << endl;
```

```
}
```

```
string getName() const {
```

```
    return name;
```

```
}
```

```
int getRollNumber() const {
```

```
    return roll_number;
```

```
}
```

```
float getMarks() const {
```

```
    return marks;
```

```
}
```

```
void setMarks(float newMarks) {
```

```
    marks = newMarks;
```

```
}
```

```
friend ostream& operator<<(ostream& os, const Student& s);
```

```
friend istream& operator>>(istream& is, Student& s);
```

```
};
```

```
ostream& operator<<(ostream& os, const Student& s) {
```

```
    os << s.name << " " << s.roll_number << " " << s.marks;
```

```
    return os;
```

```
}
```

```
istream& operator>>(istream& is, Student& s) {  
    is >> s.name >> s.roll_number >> s.marks;  
    return is;  
}
```

```
int main() {  
    Student students[5];  
    ofstream outfile("students.txt");  
    for (int i = 0; i < 5; i++) {  
        students[i].input();  
        outfile << students[i] << endl;  
    }  
    outfile.close();
```

```
    ifstream infile("students.txt");  
    for (int i = 0; i < 5; i++) {  
        infile >> students[i];  
        students[i].display();  
        cout << endl;  
    }  
    infile.close();
```

```
    fstream file("students.txt", ios::in | ios::out);  
    string targetName;  
    cout << "Enter the name of the student to modify: ";  
    cin >> targetName;
```

```

Student temp;

streampos pos;

while (file >> temp) {
    if (temp.getName() == targetName) {
        cout << "Enter new marks: ";
        float newMarks;
        cin >> newMarks;

        temp.setMarks(newMarks);

        pos = file.tellg();
        file.seekp(pos - sizeof(temp));

        file << temp.getName() << " " << temp.getRollNumber() << " " <<
temp.getMarks() << endl;
        break;
    }
}

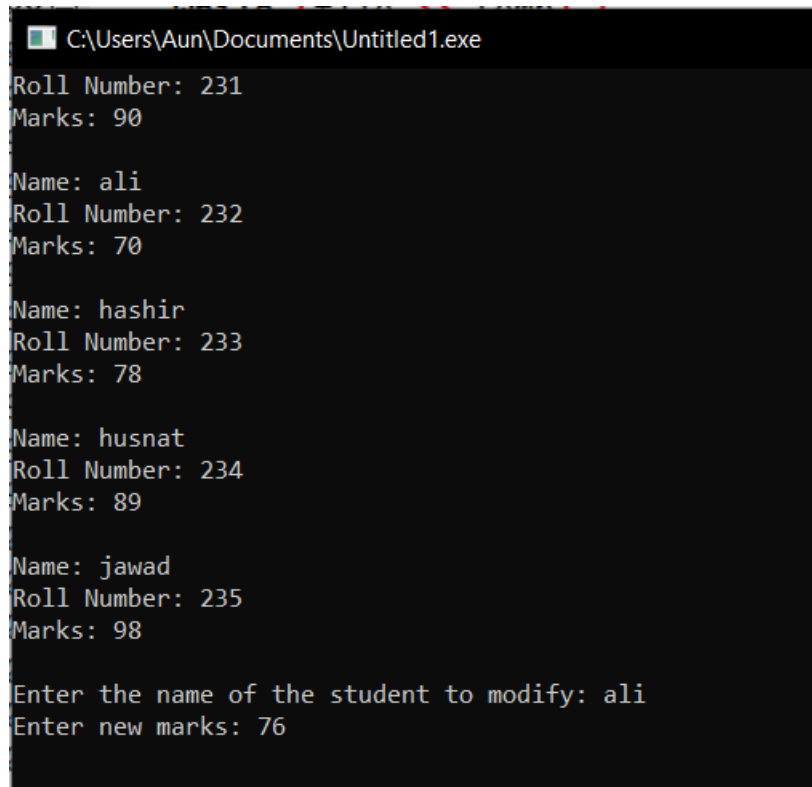
file.close();

return 0;

}

```

OUTPUT:



```
C:\Users\Aun\Documents\Untitled1.exe
Roll Number: 231
Marks: 90

Name: ali
Roll Number: 232
Marks: 70

Name: hashir
Roll Number: 233
Marks: 78

Name: husnat
Roll Number: 234
Marks: 89

Name: jawad
Roll Number: 235
Marks: 98

Enter the name of the student to modify: ali
Enter new marks: 76
```

PROGRAM 2 (A):

```
#include <iostream>
```

```
using namespace std;
```

```
class Shape {
```

```
protected:
```

```
    string color;
```

```
public:
```

```
    Shape(string c) : color(c) {}
```

```
    void setColor(string c) {
```

```

        color = c;
    }

    string getColor() {
        return color;
    }
};

class Rectangle : public Shape {
    float length, breadth;

public:
    // Constructor
    Rectangle(string c, float l, float b) : Shape(c), length(l), breadth(b) {}

    float area() {
        return length * breadth;
    }

    float perimeter() {
        return 2 * (length + breadth);
    }
};

int main() {
    Rectangle rect("Red", 5.0, 3.0);
    cout << "Color: " << rect.getColor() << endl;
}

```

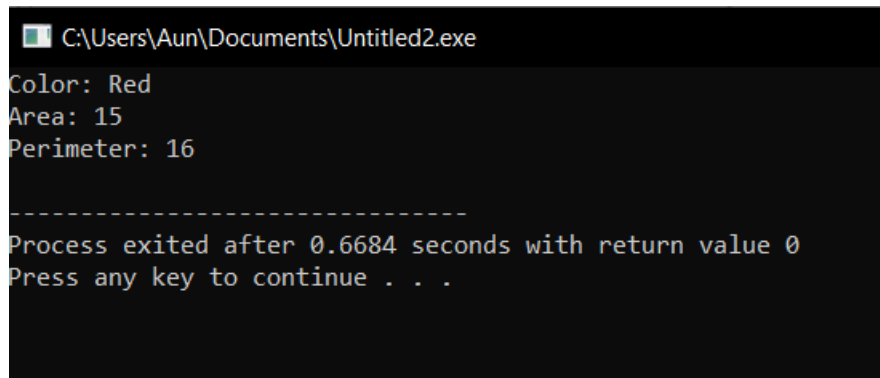
```
cout << "Area: " << rect.area() << endl;

cout << "Perimeter: " << rect.perimeter() << endl;


return 0;

}
```

OUTPUT:



```
C:\Users\Aun\Documents\Untitled2.exe
Color: Red
Area: 15
Perimeter: 16
-----
Process exited after 0.6684 seconds with return value 0
Press any key to continue . . .
```

PROGRAM 3(B):

```
#include <iostream>

using namespace std;

class Animal {
protected:
    string name;

public:
    Animal(string n) : name(n) {}
};

class Mammal : public Animal {
protected:
```



```
    int numberOfLegs;

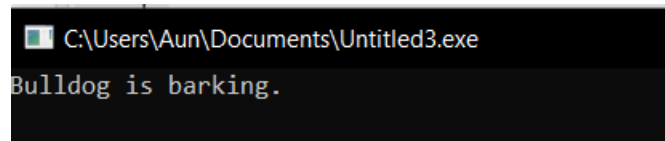
public:
    Mammal(string n, int legs) : Animal(n), numberOfLegs(legs) {}
};

class Dog : public Mammal {
public:
    Dog(string n, int legs) : Mammal(n, legs) {}

    void bark() {
        cout << name << " is barking." << endl;
    }
};

int main() {
    Dog dog("Bulldog", 4);
    dog.bark();
    return 0;
}
```

OUTPUT:



A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Users\Aun\Documents\Untitled3.exe". The command prompt shows the output "Bulldog is barking." in a monospaced font.

PROGRAM 4(C):

```
#include <iostream>

using namespace std;
```

```
class Person {
```

```
protected:
```

```
    string name;
```

```
    string address;
```

```
public:
```

```
    Person(string n, string addr) : name(n), address(addr) {}
```

```
    void displayPerson() {
```

```
        cout << "Name: " << name << endl;
```

```
        cout << "Address: " << address << endl;
```

```
    }
```

```
};
```

```
// Base class Employee
```

```
class Employee {
```

```
protected:
```

```
    int employeeID;
```

```
    float salary;
```

```
public:
```

```
Employee(int id, float sal) : employeeID(id), salary(sal) {}

void displayEmployee() {
    cout << "Employee ID: " << employeeID << endl;
    cout << "Salary: " << salary << endl;
}
};
```

```
class Teacher : public Person, public Employee {
    string subject;

public:
    Teacher(string n, string addr, int id, float sal, string subj)
        : Person(n, addr), Employee(id, sal), subject(subj) {}

    void displayTeacher() {
        displayPerson(); // Access members of Person
        displayEmployee(); // Access members of Employee
        cout << "Subject: " << subject << endl;
    }
};
```

```
int main() {
    // Creating an object of the derived class Teacher
    Teacher t("ADNAN JELANI", "F8 MARKAZ Main St", 101, 50000, "OOP");
```

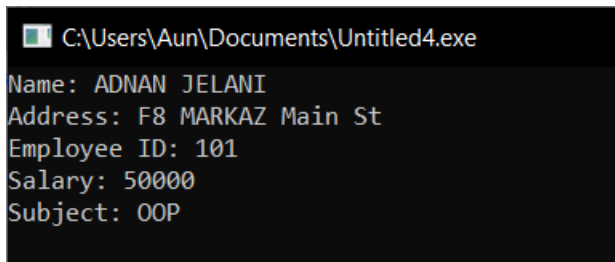
```
// Accessing members of all classes

t.displayTeacher();

return 0;

}
```

OUTPUT:



```
C:\Users\Aun\Documents\Untitled4.exe
Name: ADNAN JELANI
Address: F8 MARKAZ Main St
Employee ID: 101
Salary: 50000
Subject: OOP
```

AMBIGUITY:

If both the Person and Employee classes had a method with the same name, for example, display(), the compiler would face ambiguity when trying to resolve which display() method to call when it is invoked from the Teacher class. This is known as the ambiguity problem.

Resolving the Ambiguity Problem:

To resolve this, you can use the scope resolution operator (::) to specify which base class method you want to call.

```
class Teacher : public Person, public Employee {

    string subject;

public:

    Teacher(string n, string addr, int id, float sal, string subj)

        : Person(n, addr), Employee(id, sal), subject(subj) {}

    void display() {

        Person::display(); // Explicitly calling Person's display method
```

```
Employee::display(); // Explicitly calling Employee's display method
cout << "Subject: " << subject << endl;
}
};
```