Name: Fizaan Ali Shafiq Mughal

Roll no: 555

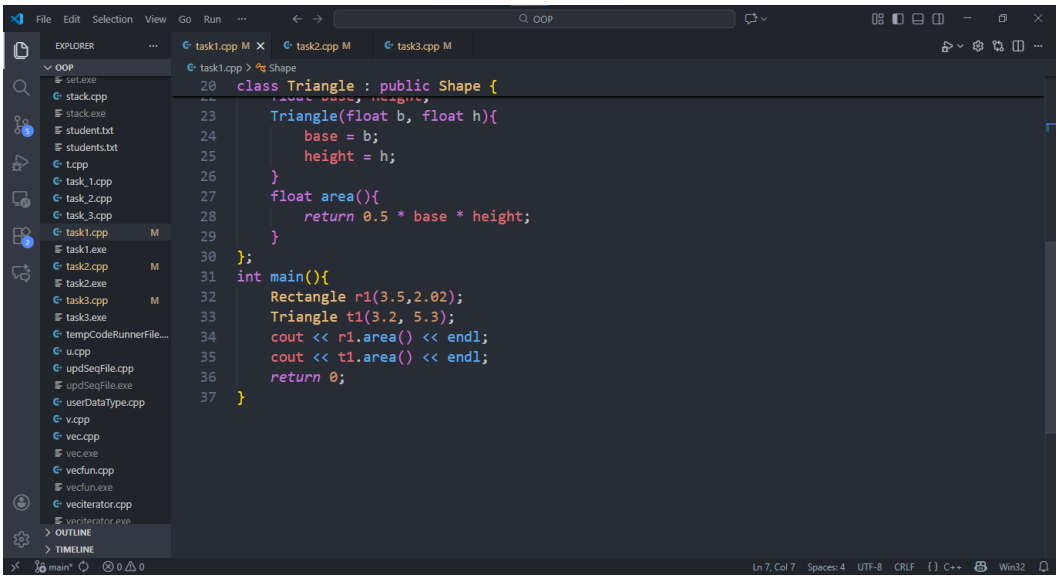
Submitted to: Mam Mahnoor

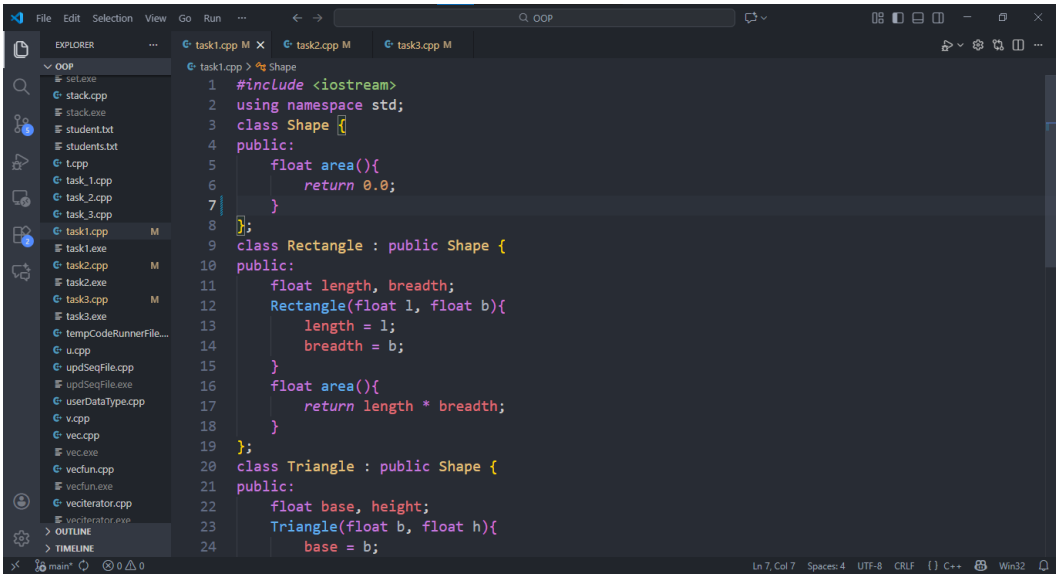
Course Code: CC-211

ASSIGNMENT NO. 2

TASK 1:

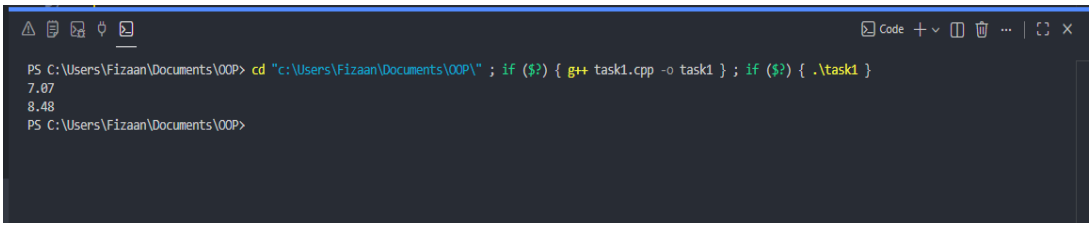
Calculate Area· Create a base class called Shape with a public member function area() that returns 0 · Then, create two subclasses called Rectangle and Triangle that inherit from the Shape class · Override the area() function in both subclasses to calculate and return the area of a rectangle and a triangle, respectively. · Finally, create instances of both the Rectangle and Triangle classes, and call the area() function on each instance to verify that the correct area is calculated and returned for each shape.





```cpp
#include <iostream>
using namespace std;
class Shape {
public:
    float area(){
        return 0.0;
    }
};
class Rectangle : public Shape {
public:
    float length, breadth;
    Rectangle(float l, float b){
        length = l;
        breadth = b;
    }
    float area(){
        return length * breadth;
    }
};
class Triangle : public Shape {
public:
    float base, height;
    Triangle(float b, float h){
        base = b;
```

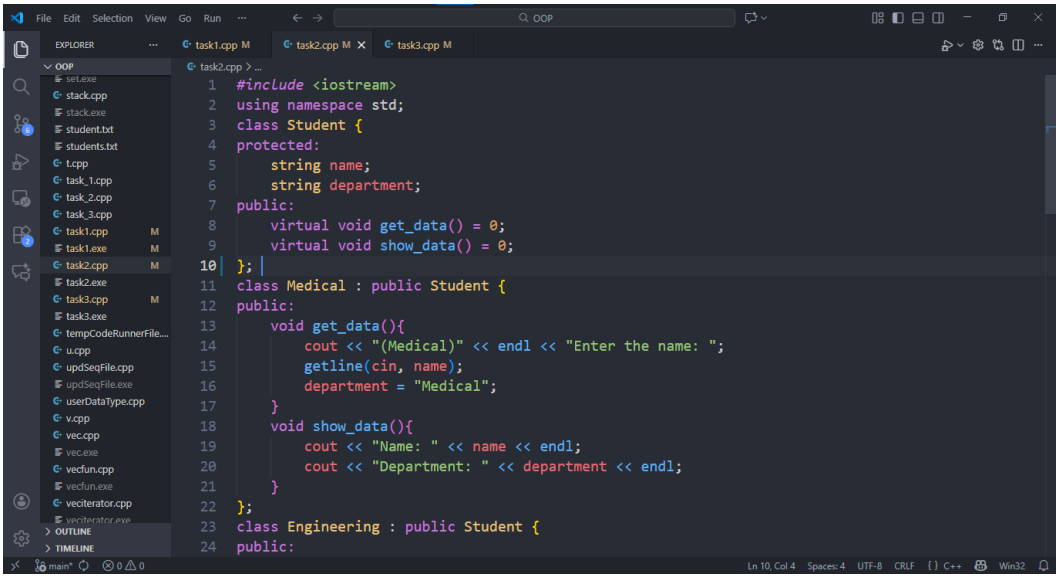
OUTPUT:



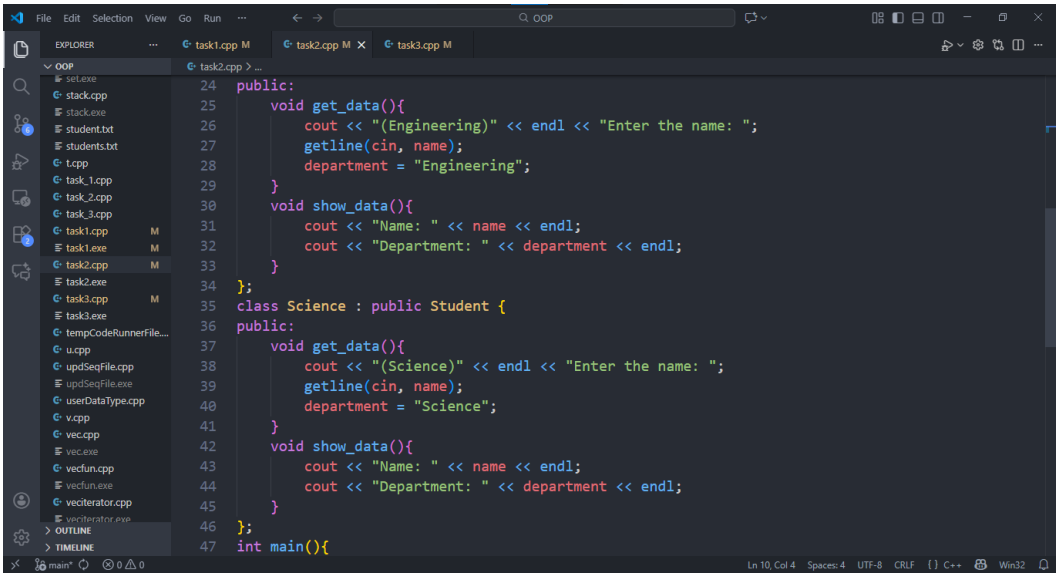
```
PS C:\Users\Fizaan\Documents\OOP> cd "c:\Users\Fizaan\Documents\OOP\" ; if ($?) { g++ task1.cpp -o task1 } ; if ($?) { .\task1 }
7.07
8.48
PS C:\Users\Fizaan\Documents\OOP>
```

TASK 2:

Student Record By implementing pure virtual function, make a class ‘Student’ with data members o name o department as protected type and two pure virtual member functions o get_data() "to input the student record" o show_data() "display student record". Derive three classes ‘Medical’, ‘Engineering’, and ‘Science’ from Student class with its own member functions of get_data() "to input the student record" and show_data() "display student record" while using data members of student class. In main create an array of pointers of base class having size three and assign to each index remaining class objects addresses. By using for loop iterate array and take input then again display records using for loop.



```cpp
#include <iostream>
using namespace std;
class Student {
protected:
    string name;
    string department;
public:
    virtual void get_data() = 0;
    virtual void show_data() = 0;
};
class Medical : public Student {
public:
    void get_data(){
        cout << "(Medical)" << endl << "Enter the name: ";
        getline(cin, name);
        department = "Medical";
    }
    void show_data(){
        cout << "Name: " << name << endl;
        cout << "Department: " << department << endl;
    }
};
class Engineering : public Student {
public:
```



```cpp
public:
    void get_data(){
        cout << "(Engineering)" << endl << "Enter the name: ";
        getline(cin, name);
        department = "Engineering";
    }
    void show_data(){
        cout << "Name: " << name << endl;
        cout << "Department: " << department << endl;
    }
};
class Science : public Student {
public:
    void get_data(){
        cout << "(Science)" << endl << "Enter the name: ";
        getline(cin, name);
        department = "Science";
    }
    void show_data(){
        cout << "Name: " << name << endl;
        cout << "Department: " << department << endl;
    }
};
int main(){
```

```cpp
    void get_data(){
        cout << "(Science)" << endl << "Enter the name: ";
        getline(cin, name);
        department = "Science";
    }
    void show_data(){
        cout << "Name: " << name << endl;
        cout << "Department: " << department << endl;
    }
};
int main(){
    Student * ptr[3];
    ptr[0] = new Medical;
    ptr[2] = new Engineering;
    ptr[1] = new Science;

    for(int i=0; i<3; i++){
        ptr[i] -> get_data();
    }
    for(int i=0; i<3; i++){
        ptr[i] -> show_data();
    }
    return 0;
}
```

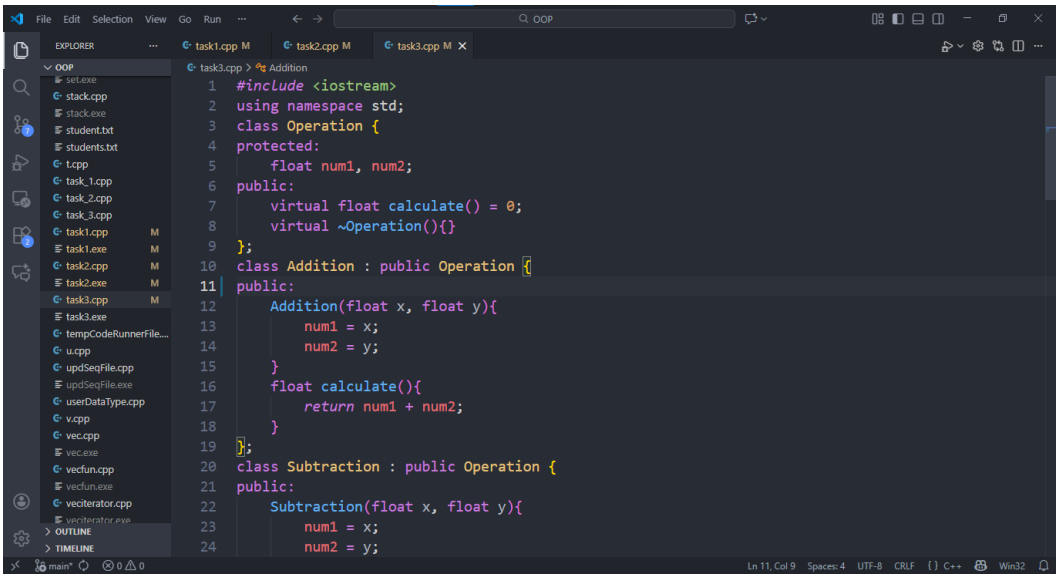
OUTPUT:

```
    at Microsoft.PowerShell.PSConsoleReadLine.Insert(Char c)
    at Microsoft.PowerShell.PSConsoleReadLine.SelfInsert(Nullable`1 key, Object arg)
    at Microsoft.PowerShell.PSConsoleReadLine.ProcessOneKey(ConsoleKeyInfo key, Dictionary`2 dispatchTable, Boolean ignoreIfNoAction, Object arg)
    at Microsoft.PowerShell.PSConsoleReadLine.InputLoop()
    at Microsoft.PowerShell.PSConsoleReadLine.ReadLine(Runspace runspace, EngineIntrinsics engineIntrinsics)
--------------------------------------------------------------------------------
PS C:\Users\Fizaan\Documents\OOP> cd "c:\Users\Fizaan\Documents\OOP\" ; if ($?) { g++ task2.cpp -o task2 } ; if ($?) { .\task2 }
(Medical)
Enter the name: Fizaan
(Science)
Enter the name: Ali
(Engineering)
Enter the name: Ans
Name: Fizaan
Department: Medical
Name: Ali
Department: Science
Name: Ans
Department: Engineering
PS C:\Users\Fizaan\Documents\OOP>
```

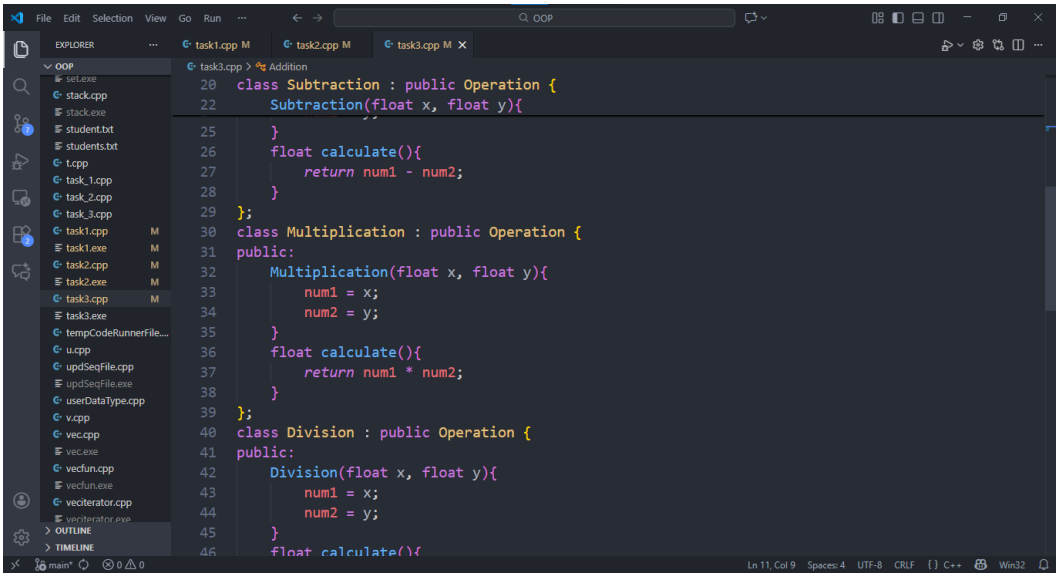
TASK 3:

Create a C++ program that simulates a simple calculator. The program should have the following features: The calculator should be able to perform basic arithmetic operations: addition, subtraction, multiplication, and division. The program should allow the user to enter two numbers and choose an operation. The program should then display the result of the operation. Finally, the program should exit when the user chooses to quit. To demonstrate the use of virtual destructor, you can create a base class called ‘Operation’, which has virtual destructor. Then, create four derived classes which inherit from Operation and implement their respective operations. The four classes are: o Addition o Subtraction o Multiplication o Division When the user chooses an operation, the program should create an object of the corresponding derived class and call its calculate() method.

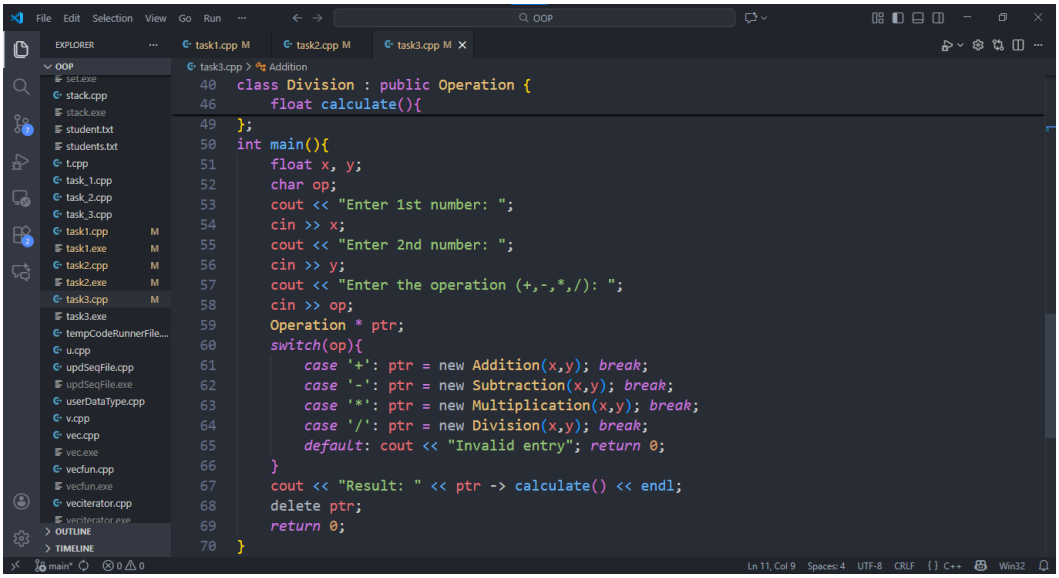
Since Operation has a virtual destructor, the destructor of the derived class will be called automatically when the object goes out of scope.



```cpp
#include <iostream>
using namespace std;
class Operation {
protected:
    float num1, num2;
public:
    virtual float calculate() = 0;
    virtual ~Operation(){}
};
class Addition : public Operation {
public:
    Addition(float x, float y){
        num1 = x;
        num2 = y;
    }
    float calculate(){
        return num1 + num2;
    }
};
class Subtraction : public Operation {
public:
    Subtraction(float x, float y){
        num1 = x;
        num2 = y;
```



```cpp
class Subtraction : public Operation {
    Subtraction(float x, float y){
    }
    float calculate(){
        return num1 - num2;
    }
};
class Multiplication : public Operation {
public:
    Multiplication(float x, float y){
        num1 = x;
        num2 = y;
    }
    float calculate(){
        return num1 * num2;
    }
};
class Division : public Operation {
public:
    Division(float x, float y){
        num1 = x;
        num2 = y;
    }
    float calculate(){
```



```cpp
class Division : public Operation {
    float calculate(){
};
int main(){
    float x, y;
    char op;
    cout << "Enter 1st number: ";
    cin >> x;
    cout << "Enter 2nd number: ";
    cin >> y;
    cout << "Enter the operation (+,-,*,/): ";
    cin >> op;
    Operation * ptr;
    switch(op){
        case '+': ptr = new Addition(x,y); break;
        case '-': ptr = new Subtraction(x,y); break;
        case '*': ptr = new Multiplication(x,y); break;
        case '/': ptr = new Division(x,y); break;
        default: cout << "Invalid entry"; return 0;
    }
    cout << "Result: " << ptr -> calculate() << endl;
    delete ptr;
    return 0;
}
```

OUTPUT:

```
PS C:\Users\Fizaan\Documents\OOP> cd "c:\Users\Fizaan\Documents\OOP\" ; if ($?) { g++ task3.cpp -o task3 } ; if ($?) { .\task3 }
Enter 1st number: 8
Enter 2nd number: 2
Enter the operation (+,-,*,/): /
Result: 4
PS C:\Users\Fizaan\Documents\OOP>
```