|  |  |
| --- | --- |
| Ex.No.**5**  **25.03.2022** | **Classes and Objects** |

**Name- Shorya shokanda Regn No.- 21BCE10310**

|  |
| --- |
| **AIM:** |

To write and execute C++ programs for the problem statements based Classes and Objects.

|  |
| --- |
| **1a. Create a class named 'Student' with a string variable 'name' and an integer variable 'roll\_no'. Assign the value of roll\_no as '2' and that of name as "John" by creating an object of the class Student.** |

***C++ Program:***

#include <iostream>

#include <string>

using namespace std;

class student

{

public:

    string name;

    int roll\_no;

};

int main()

{

    student st1;

    st1.name = "John";

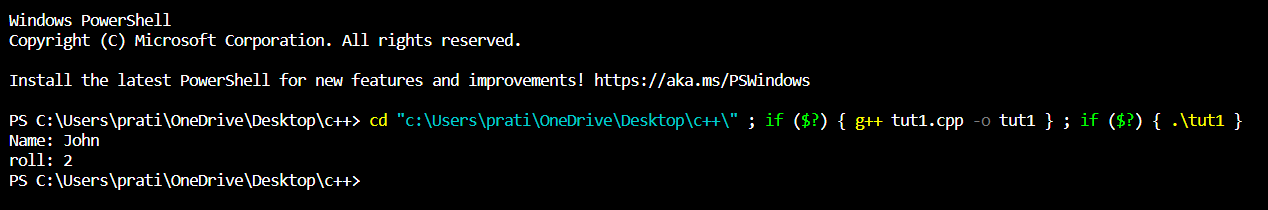
    st1.roll\_no = 2;

    cout << "Name: " << st1.name << endl;

    cout << "roll: " << st1.roll\_no << endl;

}

***Output Screenshots:***

****

|  |
| --- |
| **1b. Assign and print the roll number, phone number and address of two students having names "Sam" and "John" respectively by creating two objects of the class 'Student'.** |

***C++ Program:***

#include <iostream>

#include <string>

using namespace std;

class Student

{

    string name;

    int rollno;

    string phone;

    string address;

public:

    Student() {}

    Student(string \_name, int \_rollno, string \_phone, string \_address)

        : name(\_name), rollno(\_rollno), phone(\_phone), address(\_address) {}

    void AssignAtr(string \_name, int \_rollno, string \_phone, string \_address)

    {

        name = \_name;

        rollno = \_rollno;

        phone = \_phone;

        address = \_address;

    }

    void Print()

    {

        cout << "\nStudent`s name is\t" << name

             << "\nRoll number is\t\t" << rollno

             << "\nPhone is\t\t" << phone

             << "\nAddress is\t\t" << address;

    }

};

int main()

{

    Student s("Sam", 12, "12345678", "Central avenue, 28");

    Student j;

    j.AssignAtr("John", 10, "98765432", "Apple street, 17");

    cout << "    First student";

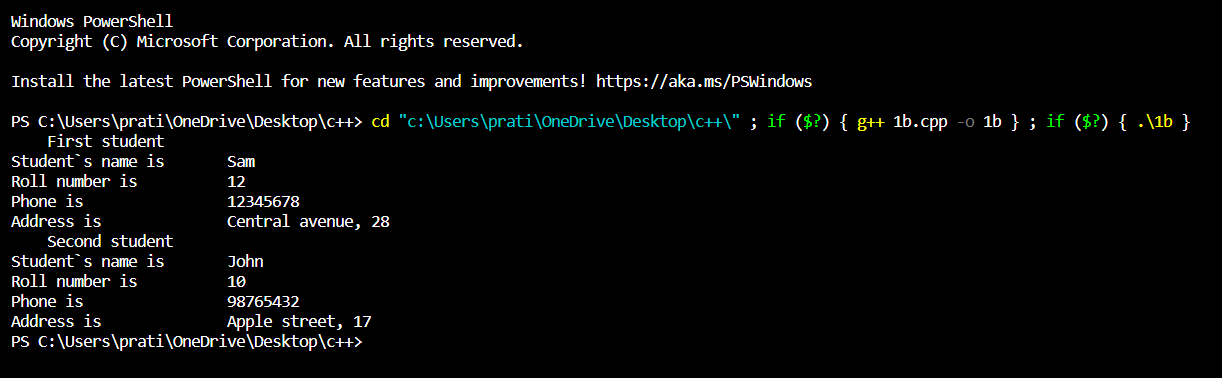
    s.Print();

    cout << "\n    Second student";

    j.Print();

}

***Output Screenshots:***

****

|  |
| --- |
| **2a. Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with a function to print the area and perimeter.** |

***C++ Program:***

#include <iostream>

#include <math.h>

using namespace std;

class Triangle

{

public:

    void area(int a, int b, int c);

    void perimeter(int a, int b, int c);

};

void Triangle::area(int a, int b, int c)

{

    float s = ((a + b + c) / 2.0), A = sqrt(s \* (s - a) \* (s - b) \* (s - c));

    cout << "Area of a triangle is: " << A << " sq. units\n";

}

void Triangle::perimeter(int a, int b, int c)

{

    cout << "Perimeter of a triangle is: " << a + b + c << " units\n";

}

int main()

{

    Triangle tr1;

    int side1 = 3, side2 = 4, side3 = 5;

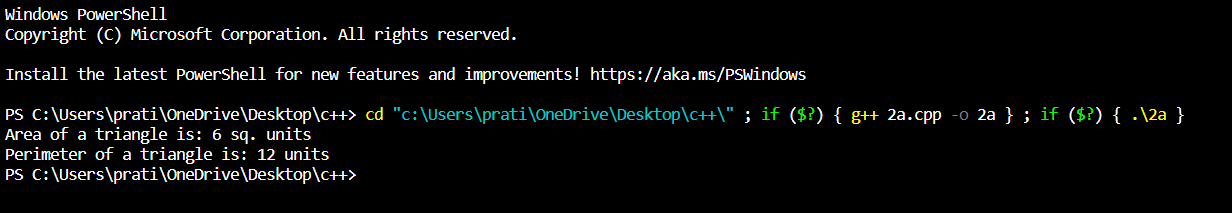
    tr1.area(side1, side2, side3);

    tr1.perimeter(side1, side2, side3);

    return 0;

}

***Output Screenshots:***

****

|  |
| --- |
| **2b. Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with the constructor having the three sides as its parameters.** |

***C++ Program:***

#include <iostream>

#include <math.h>

using namespace std;

class Triangle

{

private:

    double a, b, c;

public:

    Triangle(double a, double b, double c)

    {

        this->a = a;

        this->b = b;

        this->c = c;

    }

    double area()

    {

        double s = (a + b + c) / 2;

        return sqrt(s \* (s - a) \* (s - b) \* (s - c));

    }

    double perimeter()

    {

        return a + b + c;

    }

};

// The start point of the program

int main()

{

    double a, b, c;

    cout << "Ente a: ";

    cin >> a;

    cout << "Ente b: ";

    cin >> b;

    cout << "Ente c: ";

    cin >> c;

    Triangle triangle(a, b, c);

    cout << "Area: " << triangle.area() << "\n";

    cout << "Perimeter: " << triangle.perimeter() << "\n";

    system("pause");

    return 0;

}

***Output Screenshots:***



|  |
| --- |
| **3a. Write a program to print the area of two rectangles having sides (4,5) and (5,8) respectively by creating a class named 'Rectangle' with a function named 'Area' which returns the area. Length and breadth are passed as parameters to its constructor.** |

***C++ Program:***

#include <iostream>

using namespace std;

class Rect

{

private:

    int a, b;

public:

    Rect(int a, int b)

    {

        this->a = a;

        this->b = b;

    }

    int area()

    {

        return this->a \* this->b;

    }

};

int main()

{

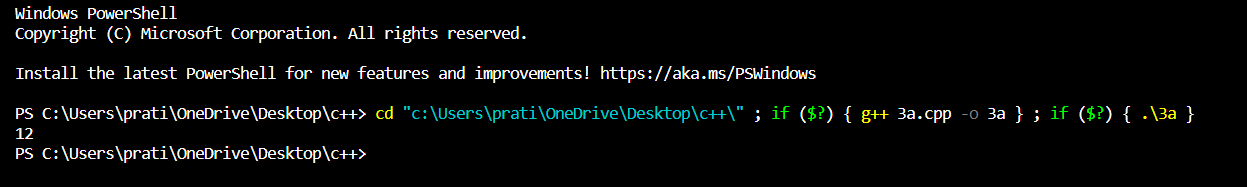
    Rect obj = Rect(3, 4);

    cout << obj.area() << "\n";

    return 0;

}

***Output Screenshots:***



|  |
| --- |
| **Using Switch Case Statements** |

|  |
| --- |
| **3b. Write a program to print the area of a rectangle by creating a class named 'Area' having two functions. First function named as 'setDim' takes the length and breadth of the rectangle as parameters and the second function named as 'getArea' returns the area of the rectangle. Length and breadth of the rectangle are entered through keyboard.** |

***C++ Program:***

#include <iostream>

using namespace std;

class area

{

public:

    int length = 0, breadth = 0;

    void setdim(int l, int b)

    {

        length = l;

        breadth = b;

    }

    int getarea()

    {

        return length \* breadth;

    }

};

int main()

{

    area a1;

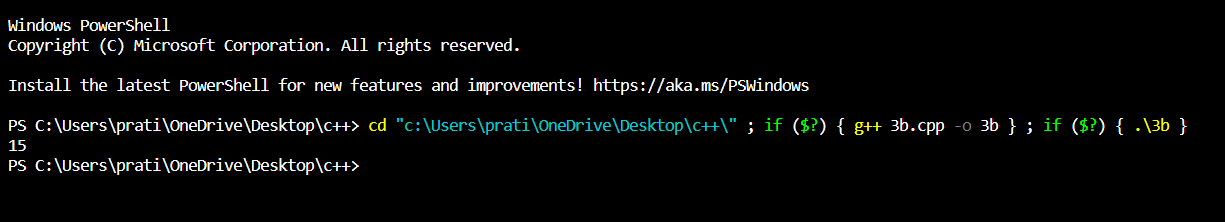
    a1.setdim(3, 5);

    cout << a1.getarea() << endl;

    return 0;

}

***Output Screenshots:***



|  |
| --- |
| **3c. Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a function named 'returnArea' which returns the area of the rectangle. Length and breadth of the rectangle are entered through keyboard.** |

***C++ Program:***

#include <iostream>

using namespace std;

class Area

{

public:

    int len, width;

    void getArea()

    {

        cout << "Enter the length: ";

        cin >> len;

        cout << "Enter width: ";

        cin >> width;

    }

    void returnArea()

    {

        cout << "Area is: " << len \* width;

    }

};

int main()

{

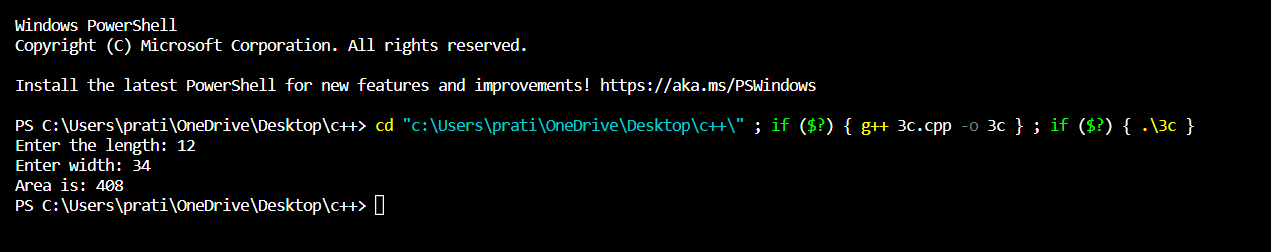
    Area A;

    A.getArea();

    A.returnArea();

}

***Output Screenshots:***



|  |
| --- |
| **4. Print the average of three numbers entered by the user by creating a class named 'Average' having a function to calculate and print the average without creating any object of the Average class.** |

***C++ Program:***

#include <iostream>

using namespace std;

class Average

{

public:

    static float calcAverate(float a, float b, float c)

    {

        return (a + b + c) / 3;

    }

};

int main()

{

    cout << "Enter three numbers: ";

    float a, b, c;

    cin >> a;

    cin >> b;

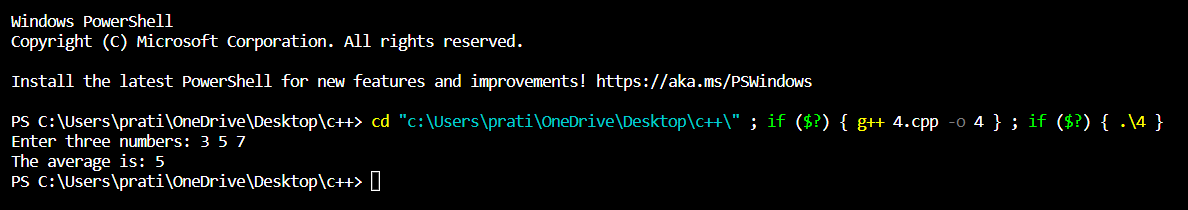
    cin >> c;

    cout << "The average is: " << Average::calcAverate(a, b, c) << endl;

    return 0;

}

***Output Screenshots:***

******

|  |
| --- |
| **5. Print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate functions for each operation whose real and imaginary parts are entered by the user.** |

***C++ Program:***

#include <iostream>

using namespace std;

class Complex

{

    double re;

    double im;

public:

    // Default constructor

    Complex() {}

    // Init constructor

    Complex(double \_re, double \_im)

        : re(\_re), im(\_im) {}

    // Copy constructor

    Complex(Complex &x)

    {

        re = x.re;

        im = x.im;

    }

    Complex operator+(Complex &x)

    {

        re = re + x.re;

        im = im + x.im;

        return \*this;

    }

    Complex operator-(Complex &x)

    {

        re = re - x.re;

        im = im - x.im;

        return \*this;

    }

    Complex operator\*(Complex &x)

    {

        re = re \* x.re - im \* x.im;

        im = re \* x.im + x.re \* im;

        return \*this;

    }

    friend ostream &operator<<(ostream &, Complex &);

    friend istream &operator>>(istream &, Complex &);

    //  friend Complex operator+(Complex&,Complex&);

};

istream &operator>>(istream &is, Complex &x)

{

    cout << "Please, enter a real part of complex number: ";

    is >> x.re;

    cout << "Please, enter an imaginary part of complex number: ";

    is >> x.im;

    return is;

}

ostream &operator<<(ostream &os, Complex &x)

{

    os << x.re;

    if (x.im > 0)

    {

        os << "+" << x.im << "i";

    }

    else if (x.im < 0)

    {

        os << x.im << "i";

    }

    return os;

}

int main()

{

    Complex a, b;

    cin >> a;

    cin >> b;

    cout << "You entered two complex numbers:\n";

    cout << "a= " << a

         << "\nb= " << b << endl;

    a + b;

    cout << "a+b= " << a << endl;

    a - b;

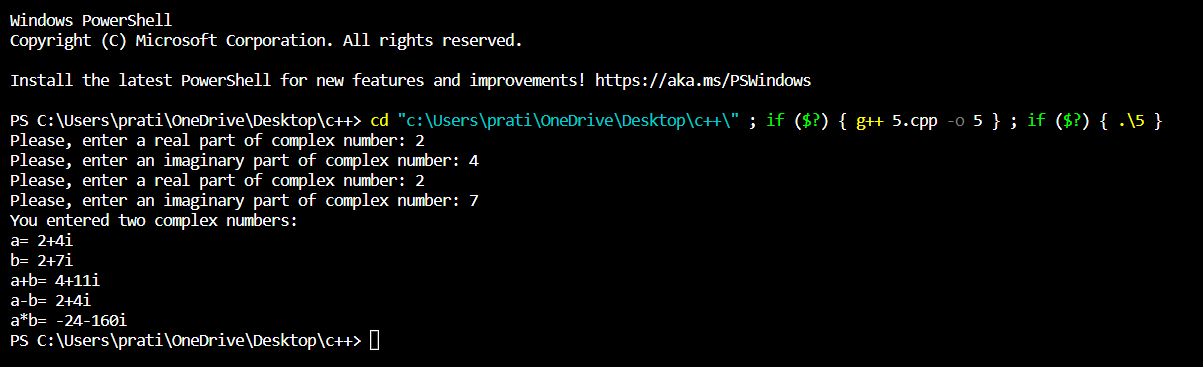
    cout << "a-b= " << a << endl;

    a \*b;

    cout << "a\*b= " << a << endl;

}

***Output Screenshots:***



|  |
| --- |
| **6. Write a program to print the volume of a box by creating a class named 'Volume' with an initialization list to initialize its length, breadth and height. (just to make you familiar with initialization lists)** |

***C++ Program:***

#include <iostream>

using namespace std;

class box

{

    double length, width, height;

    double volume;

public:

    box(double a, double b, double c);

    void vol();

};

box::box(double a, double b, double c)

{

    length = a;

    width = b;

    height = c;

    volume = length \* width \* height;

}

void box::vol()

{

    cout << "\nDimensions of Box are :: \n";

    cout << "\nLength of Box :: " << length << "\n";

    cout << "\nWidth of Box :: " << width << "\n";

    cout << "\nHeight of Box :: " << height << "\n";

    cout << "\nVolume of Box :: " << volume << "\n";

}

int main()

{

    box x(2.4, 5.7, 2.1), y(3.3, 4.4, 5.5);

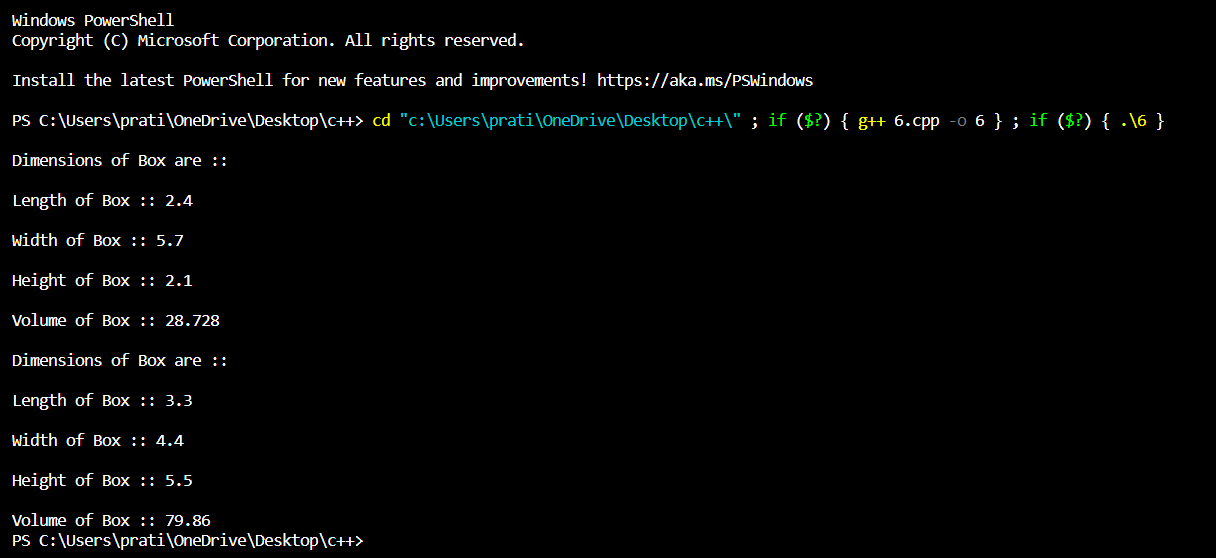
    x.vol();

    y.vol();

    return 0;

}

***Output Screenshots:***

****

|  |
| --- |
| **7. Write a program that would print the information (name, year of joining, salary, address) of three employees by creating a class named 'Employee' (using array of objects). The output should be as follows:**  **Name Year of joining Address**  **Robert 1994 64C- WallsStreat**  **Sam 2000 68D- WallsStreat**  **John 1999 26B- WallsStreat** |

***C++ Program:***

#include <iostream>

using namespace std;

class Employee

{

public:

    string name;

    string joiningyear;

    string salary;

    string address;

    Employee(string n, string j, string s, string a)

    {

        name = n;

        joiningyear = j;

        salary = s;

        address = a;

    }

    void print()

    {

        cout << name << "\t" << joiningyear << "\t\t\t" << address << endl;

    }

};

int main()

{

    Employee e1("Robert", "1994", "100000", "64C- WallStreat");

    Employee e2("Sam", "2000", "200000", "68-D WallStreat");

    cout << "Name\t Year of joining\t Address" << endl;

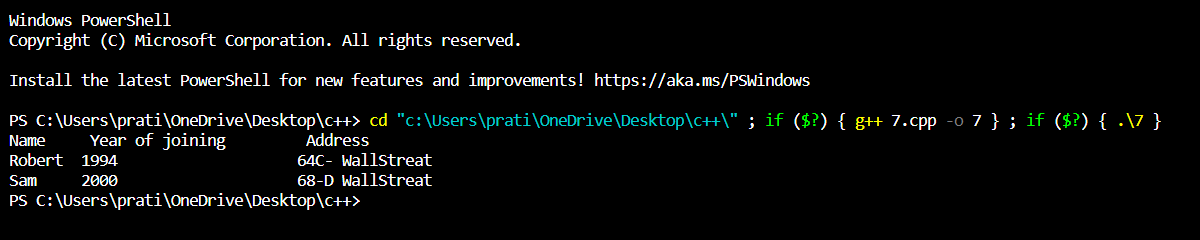
    e1.print();

    e2.print();

    return 0;

}

***Output Screenshots:***

****

|  |
| --- |
| **8. Add two distances in inch-feet by creating a class named 'AddDistance'.**  **A**  **B B**  **C C C**  **D D D D**  **E E E E E** |

***C++ Program:***

#include <iostream>

using namespace std;

class AddDistance

{

private:

    int feet;

    int inch;

public:

    void setDistance();

    void getDistance();

    AddDistance addDistance(AddDistance d);

};

// function to show print distance

void AddDistance::setDistance()

{

    cout << " feet: ";

    cin >> feet;

    cout << "inches: ";

    cin >> inch;

}

// function to get distance from user

void AddDistance::getDistance()

{

    cout << "feet: " << feet;

    cout << " inches: " << inch;

}

// function to add 2 distances

AddDistance AddDistance::addDistance(AddDistance d)

{

    AddDistance dist;

    dist.feet = feet + d.feet;

    dist.inch = inch + d.inch;

    if (dist.inch >= 12)

    {

        dist.feet++;

        dist.inch -= 12;

    }

    return dist;

}

int main()

{

    AddDistance d1, d2, d3;

    // taking the two distances as input from user

    cout << "Enter length of Distance 1: " << endl;

    d1.setDistance();

    cout << "Enter length of Distance 2: " << endl;

    d2.setDistance();

    // adding the distances

    d3 = d1.addDistance(d2);

    // displaying resultant distance

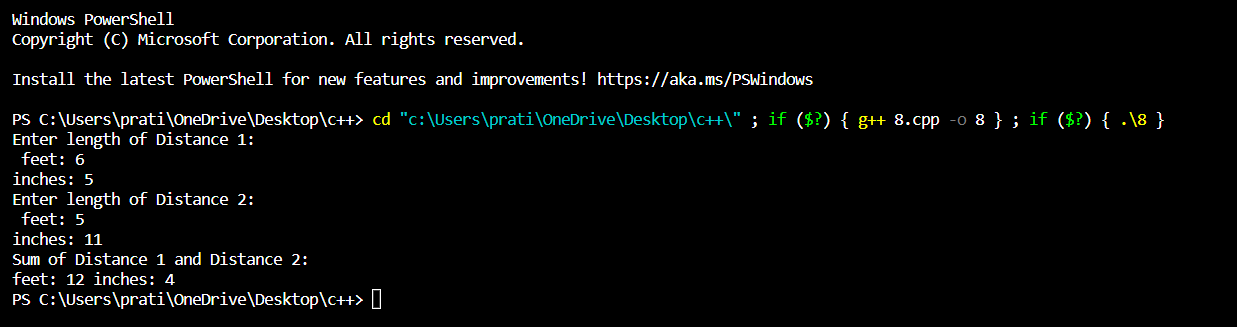
    cout << "Sum of Distance 1 and Distance 2:" << endl;

    d3.getDistance();

    return 0;

}

***Output Screenshots:***

****

|  |
| --- |
| **9. Write a program by creating an 'Employee' class having the following functions and print the final salary.**  **1 - 'getInfo()' which takes the salary, number of hours of work per day of employee as parameters**  **2 - 'AddSal()' which adds $10 to the salary of the employee if it is less than $500.**  **3 - 'AddWork()' which adds $5 to the salary of the employee if the number of hours of work per day is more than 6 hours.** |

***C++ Program:***

#include <iostream>

using namespace std;

class Employee

{

    double salary;

    int no\_of\_hours;

public:

    Employee() {}

    void getinfo()

    {

        cout << "Please, enter the salary of employee: ";

        cin >> salary;

        cout << "Please, enter the number of hours: ";

        cin >> no\_of\_hours;

    }

    void AddSal()

    {

        if (salary < 500)

            salary += 10;

    }

    void AddWork()

    {

        if (no\_of\_hours > 6)

            salary += 5;

    }

    void DisplaySalary()

    {

        cout << salary;

    }

};

int main()

{

    int num;

    cout << "Enter the number of employees: ";

    cin >> num;

    Employee \*emp = new Employee[num];

    for (int i = 0; i < num; i++)

    {

        emp[i].getinfo();

        emp[i].AddSal();

        emp[i].AddWork();

    }

    for (int i = 0; i < num; i++)

    {

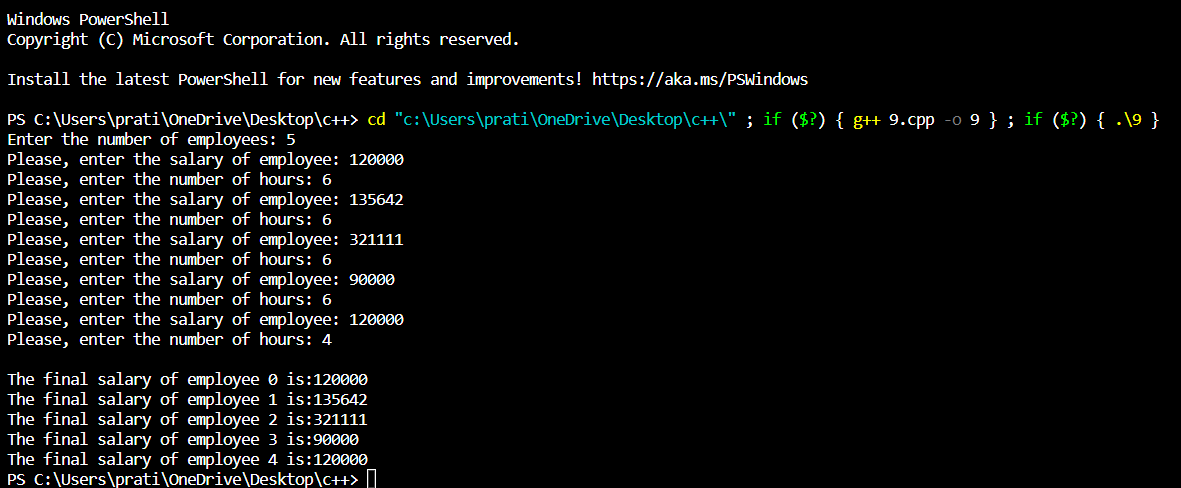
        cout << "\nThe final salary of employee " << i << " is:";

        emp[i].DisplaySalary();

    }

}

***Output Screenshots:***

****

|  |
| --- |
| **10. Create a class called 'Matrix' containing constructor that initializes the number of rows and the number of columns of a new Matrix object. The Matrix class has the following information:**  **1 - number of rows of matrix**  **2 - number of columns of matrix**  **3 - elements of matrix (You can use 2D vector)**  **The Matrix class has functions for each of the following:**  **1 - get the number of rows**  **2 - get the number of columns**  **3 - set the elements of the matrix at a given position (i,j)**  **4 - adding two matrices.**  **5 - multiplying the two matrices**  **You can assume that the dimensions are correct for the multiplication and addition.** |

***C++ Program:***

#include <iostream>

using namespace std;

class Matrix

{

    int row, column;

    int \*\*matrix;

public:

    Matrix()

    {

        row = 4;

        column = 4;

        matrix = (int \*\*)new int[row];

        for (int i = 0; i < row; i++)

            matrix[i] = new int[row];

        for (int i = 0; i < row; i++)

            for (int j = 0; j < column; j++)

                matrix[i][j] = 0;

    }

    void getRow();

    void getColumn();

    void setValue(int, int);

    void add(Matrix);

    void multiply(Matrix);

};

void Matrix::getRow()

{

    cout << "The number of rows are : " << row << endl;

}

void Matrix::getColumn()

{

    cout << "The number of rows are : " << row << endl;

}

void Matrix::setValue(int i, int j)

{

    int value;

    cout << "Enter the value to be set in the cell (" << i << ", " << j << ") :";

    cin >> value;

    matrix[i - 1][j - 1] = value;

}

void Matrix::add(Matrix m1)

{ // to add two matrices

    Matrix temp;

    for (int i = 0; i < m1.row; i++)

    { // adding the matrices

        for (int j = 0; j < m1.column; j++)

            temp.matrix[i][j] = matrix[i][j] + m1.matrix[i][j];

    }

    cout << "Result of addition" << endl;

    for (int i = 0; i < m1.row; i++)

    { // displaying the matrix

        for (int j = 0; j < m1.column; j++)

            cout << temp.matrix[i][j] << " ";

        cout << endl;

    }

}

void Matrix::multiply(Matrix m1)

{ // to multiply two matrices

    Matrix temp;

    for (int i = 0; i < row; i++) // multiplying the matrices

        for (int j = 0; j < column; j++)

            for (int k = 0; k < column; k++)

            {

                temp.matrix[i][j] += (m1.matrix[i][k] \* matrix[k][j]);

            }

    cout << "Result of multiplication" << endl;

    for (int i = 0; i < m1.row; i++)

    { // displaying the matrix

        for (int j = 0; j < m1.column; j++)

            cout << temp.matrix[i][j] << " ";

        cout << endl;

    }

}

int main()

{

    Matrix m1, m2; // two matrix objects are defined.

    m1.getColumn();

    m2.getRow();

    cout << "Start setting the values of matrix m1 according to the index value.\n";

    m1.setValue(3, 3); // the value of the elements are set by the function setValue()

    m1.setValue(4, 1);

    m1.setValue(1, 1);

    m1.setValue(2, 1);

    m1.setValue(3, 1);

    m1.setValue(1, 4);

    cout << "Start setting the values of matrix m1 according to the index value.\n";

    m2.setValue(2, 3); // the value of the elements are set by the function setValue()

    m2.setValue(3, 1);

    m2.setValue(2, 4);

    m2.setValue(1, 1);

    m2.setValue(4, 1);

    m2.setValue(3, 3);

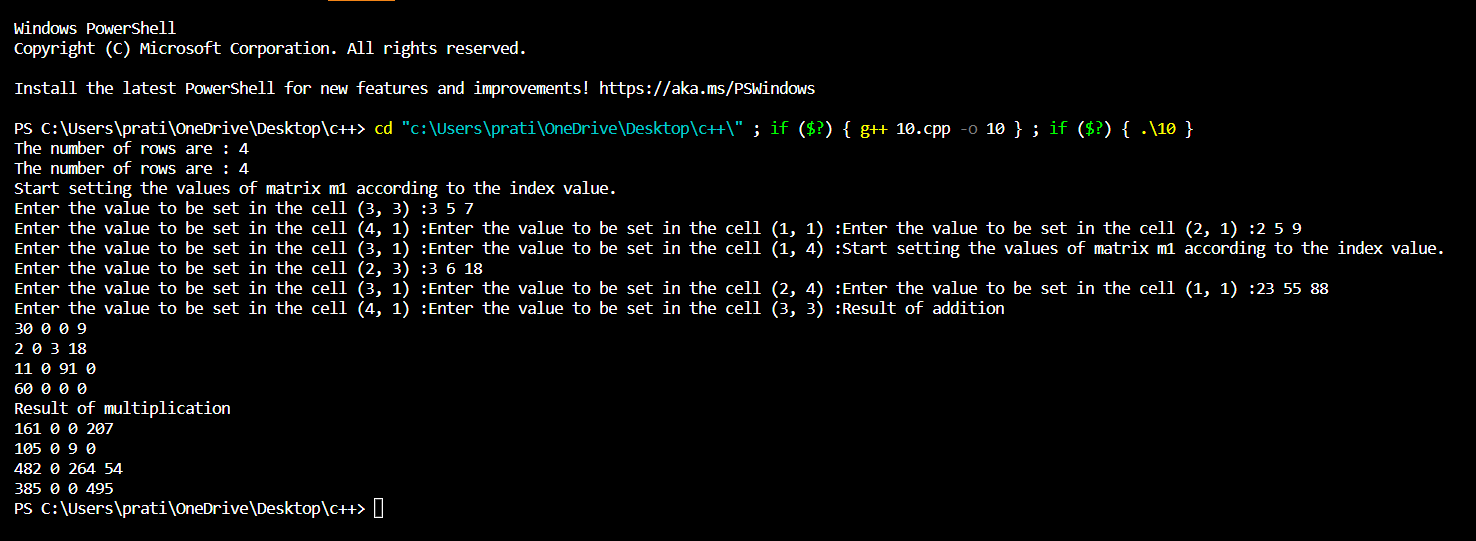
    m1.add(m2); // the two matrices are added

    m1.multiply(m2); // the two matrices are multiplied

    return 0;

}

***Output Screenshots:***



|  |
| --- |
| **RESULT:** |

Thus, the programs for the given problem statements has been executed and the results are verified successfully.