**DAY 3 LAB**

**HARD**

**1.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Student {

private:

string name;

public:

Student(string studentName = "Unknown")

{

name = studentName;

}

string getName() {

return name;

}

};

int main() {

Student student1;

Student student2("John");

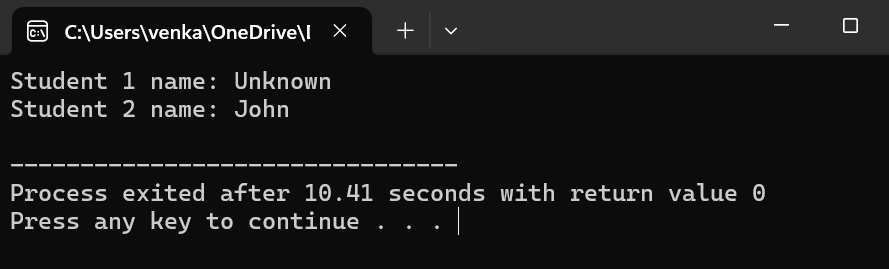
cout << "Student 1 name: " << student1.getName() << endl;

cout << "Student 2 name: " << student2.getName() << endl;

return 0;

}

**OUTPUT**

****

**2.PROGRAM**

#include <iostream>

using namespace std;

class Rectangle

{

private:

double length;

double breadth;

public:

Rectangle()

{

length = 0;

breadth = 0;

}

Rectangle(double len, double brd)

{

length = len;

breadth = brd;

}

Rectangle(double num)

{

length = num;

breadth = num;

}

double calculateArea() {

return length \* breadth;

}

};

int main() {

Rectangle rect1;

Rectangle rect2(5, 3);

Rectangle rect3(4);

cout << "Area of Rectangle 1: " << rect1.calculateArea() << endl;

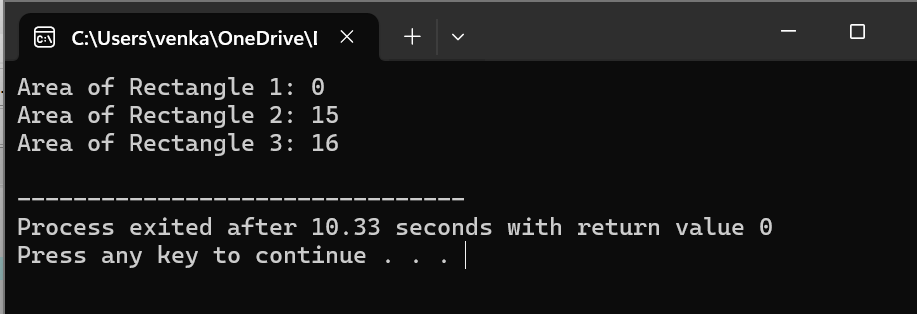
cout << "Area of Rectangle 2: " << rect2.calculateArea() << endl;

cout << "Area of Rectangle 3: " << rect3.calculateArea() << endl;

return 0;

}

**OUTPUT**



**3.PROGRAM**

#include <iostream>

using namespace std;

class AddAmount

{

private:

double amount;

public:

AddAmount()

{

amount = 50;

}

AddAmount(double additionalAmount)

{

amount = 50 + additionalAmount;

}

double getAmount()

{

return amount;

}

};

int main()

{

AddAmount piggyBank1;

AddAmount piggyBank2(30);

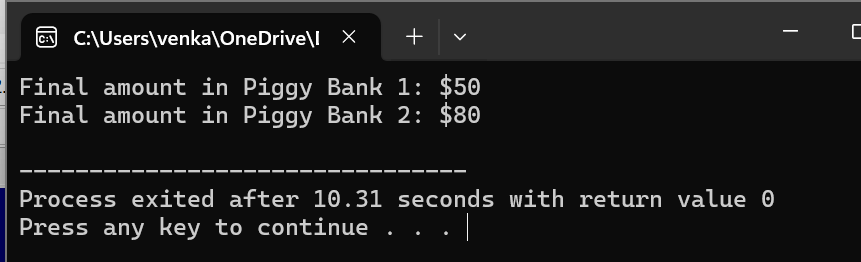
cout << "Final amount in Piggy Bank 1: $" << piggyBank1.getAmount() << endl;

cout << "Final amount in Piggy Bank 2: $" << piggyBank2.getAmount() << endl;

return 0;

}

**OUTPUT**

****

**4.PROGRAM**

#include <iostream>

using namespace std;

class AreaCalculator

{

public:

void printArea(double length, double breadth)

{

double area = length \* breadth;

cout << "Area of Rectangle: " << area << endl;

}

void printArea(double side)

{

double area = side \* side;

cout << "Area of Square: " << area << endl;

}

};

int main()

{

AreaCalculator calculator;

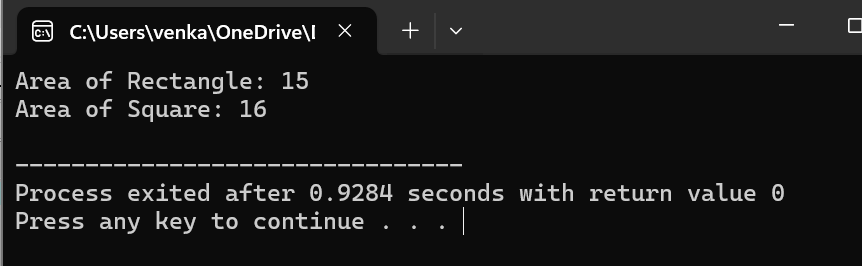
calculator.printArea(5, 3);

calculator.printArea(4);

return 0;

}

**OUTPUT**



**5.PROGRAM**

#include <iostream>

using namespace std;

class Add

{

private:

int num1;

int num2;

public:

Add(int n1, int n2) : num1(n1), num2(n2) {}

Add operator+(const Add& other)

{

Add result(num1 + other.num1, num2 + other.num2);

return result;

}

void displaySum()

{

cout << "Sum: " << num1 + num2 << endl;

}

};

int main()

{

Add obj1(5, 3);

Add obj2(8, 2);

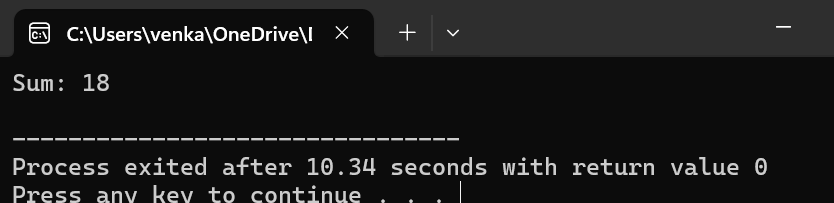
Add sum = obj1 + obj2;

sum.displaySum();

return 0;

}

**OUTPUT**



**MEDIUM**

**1.PROGRAM**

#include <iostream>

#include <cmath>

using namespace std;

class Shape

{

public:

Shape() {}

Shape(double radius)

{

area = M\_PI \* radius \* radius;

}

Shape(double length, double breadth)

{

area = length \* breadth;

}

Shape(double base, double height, char flag)

{

area = 0.5 \* base \* height;

}

void displayArea()

{

cout << "Area: " << area << endl;

}

private:

double area;

};

int main()

{

Shape circle(5);

cout << "Circle: ";

circle.displayArea();

Shape rectangle(4, 6);

cout << "Rectangle: ";

rectangle.displayArea();

Shape triangle(3, 4, 'T');

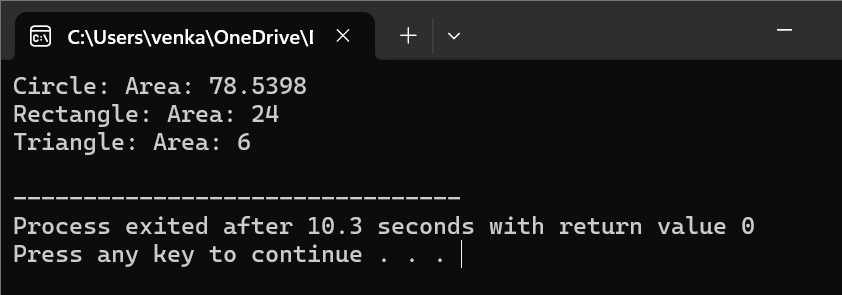
cout << "Triangle: ";

triangle.displayArea();

return 0;

}

**OUTPUT**



**2.PROGRAM**

#include <iostream>

#include <cmath>

using namespace std;

class Shape

{

public:

Shape(double sideLength)

{

volume = pow(sideLength, 3);

}

Shape(double radius, double height, char flag)

{

volume = M\_PI \* pow(radius, 2) \* height;

}

void displayVolume()

{

cout << "Volume: " << volume << endl;

}

private:

double volume;

};

int main()

{

Shape cube(5);

cout << "Cube: ";

cube.displayVolume();

Shape cylinder(3, 4, 'C');

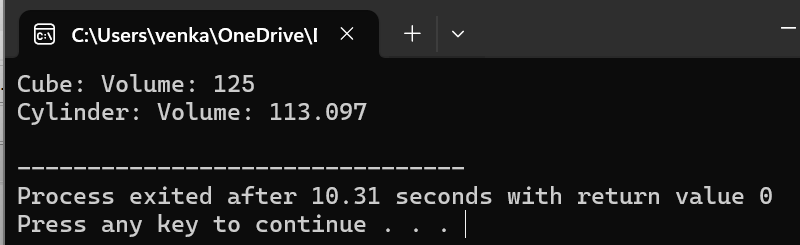
cout << "Cylinder: ";

cylinder.displayVolume();

return 0;

}

**OUTPUT**



**3.PROGRAM**

#include <iostream>

using namespace std;

class MyClass {

public:

MyClass(int num);

void display();

private:

int value;

};

MyClass::MyClass(int num)

{

value = num;

}

void MyClass::display()

{

cout << "Value: " << value << endl;

}

int main()

{

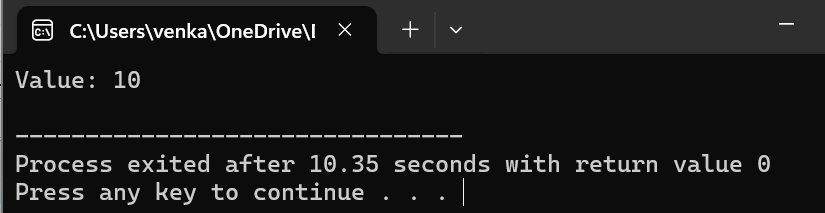
MyClass obj(10);

obj.display();

return 0;

}

**OUTPUT**



**4.PROGRAM**

#include <iostream>

using namespace std;

class FloydTriangle {

private:

int rows;

public:

FloydTriangle(int numRows)

{

rows = numRows;

}

~FloydTriangle()

{

cout << "Destructor called" << endl;

}

void printTriangle()

{

int num = 1;

for (int i = 1; i <= rows; ++i)

{

for (int j = 1; j <= i; ++j)

{

cout << num << " ";

++num;

}

cout << endl;

}

}

};

int main()

{

int numRows;

cout << "Enter the number of rows for Floyd's Triangle: ";

cin >> numRows;

FloydTriangle triangle(numRows);

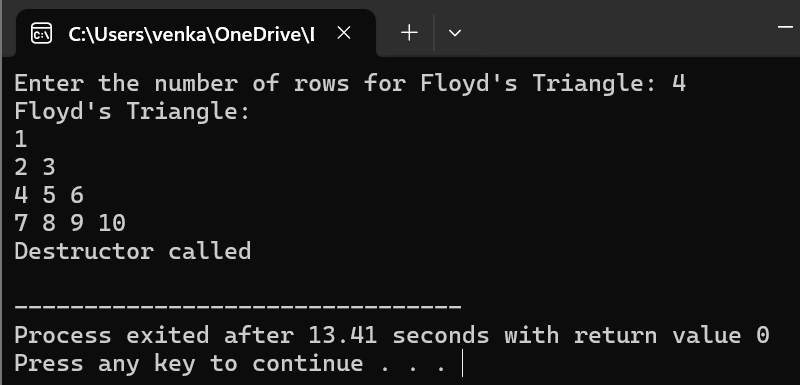
cout << "Floyd's Triangle:" << endl;

triangle.printTriangle();

return 0;

}

**OUTPUT**



**5.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class DecimalToBinary

{

private:

string binary;

public:

DecimalToBinary(int decimal)

{

binary = convertToBinary(decimal);

}

string convertToBinary(int decimal)

{

if (decimal == 0)

{

return "0";

}

string binaryResult;

while (decimal > 0)

{

binaryResult = to\_string(decimal % 2) + binaryResult;

decimal /= 2;

}

return binaryResult;

}

void displayBinary()

{

cout << "Binary representation: " << binary << endl;

}

};

int main()

{

int decimal;

cout << "Enter a decimal number: ";

cin >> decimal;

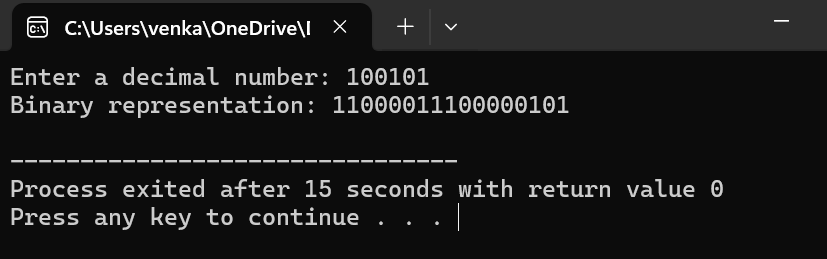
DecimalToBinary converter(decimal);

converter.displayBinary();

return 0;

}

**OUTPUT**

****

**EASY**

**1.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class DecimalToBinary

{

private:

string binary;

public:

DecimalToBinary(int decimal)

{

binary = convertToBinary(decimal);

}

string convertToBinary(int decimal)

{

if (decimal == 0)

{

return "0";

}

string binaryResult;

while (decimal > 0)

{

binaryResult = to\_string(decimal % 2) + binaryResult;

decimal /= 2;

}

return binaryResult;

}

void displayBinary()

{

cout << "Binary representation: " << binary << endl;

}

};

int main()

{

int decimal;

cout << "Enter a decimal number: ";

cin >> decimal;

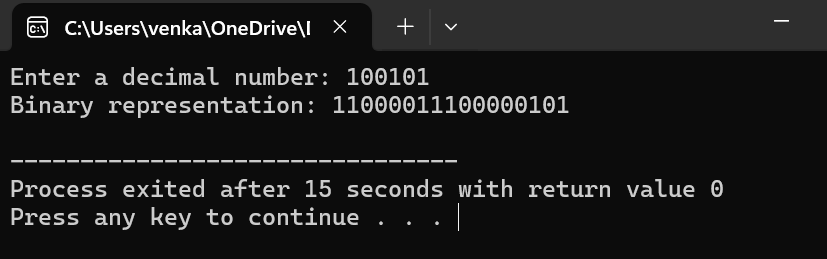
DecimalToBinary converter(decimal);

converter.displayBinary();

return 0;

}

**OUTPUT**

****

**2.PROGRAM**

#include <iostream>

using namespace std;

class CubeNumbers

{

private:

int limit;

public:

CubeNumbers(int n)

{

limit = n;

}

void displayCubes()

{

cout << "Cubes of numbers up to " << limit << ":" << endl;

for (int i = 1; i <= limit; ++i) {

cout << "Cube of " << i << ": " << i \* i \* i << endl;

}

}

};

int main()

{

int n;

cout << "Enter an integer: ";

cin >> n;

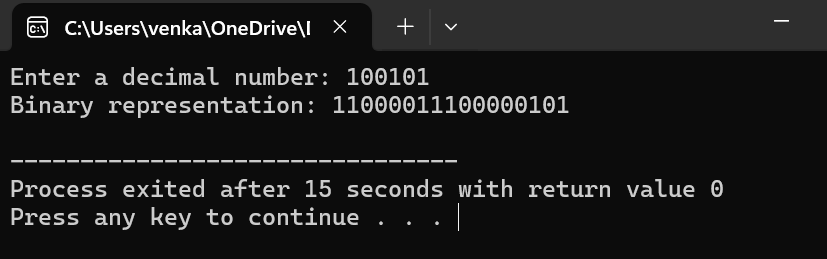
CubeNumbers cubeNumbers(n);

cubeNumbers.displayCubes();

return 0;

}

**OUTPUT**



**3.PROGRAM**

#include <iostream>

using namespace std;

class SeriesSum

{

private:

double sum;

public:

SeriesSum(int n)

{

sum = calculateSum(n);

}

SeriesSum(double x)

{

sum = calculateSum(x);

}

int calculateSum(int n)

{

return (n \* (n + 1)) / 2;

}

double calculateSum(double x)

{

return (x \* (x + 1)) / 2.0;

}

void displaySum()

{

cout << "Sum: " << sum << endl;

}

};

int main()

{

SeriesSum intSeries(5);

cout << "Sum of series of integers: ";

intSeries.displaySum();

SeriesSum floatSeries(5.0);

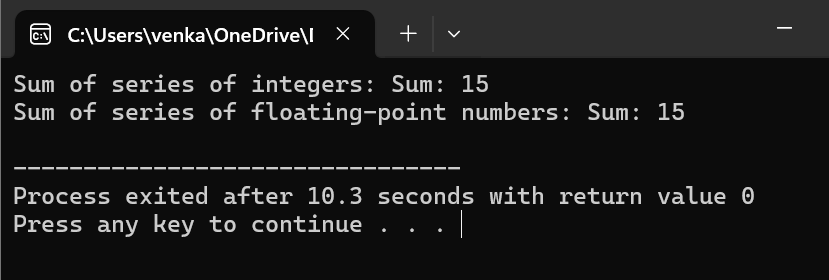
cout << "Sum of series of floating-point numbers: ";

floatSeries.displaySum();

return 0;

}

**OUTPUT**



**4.PROGRAM**

#include <iostream>

using namespace std;

class RightAngleTriangle

{

private:

int rows;

public:

RightAngleTriangle(int numRows)

{

rows = numRows;

}

void printPattern()

{

int num = 1;

for (int i = 1; i <= rows; ++i)

{

for (int j = 1; j <= i; ++j)

{

cout << num << " ";

}

cout << endl;

++num;

}

}

};

int main()

{

int numRows;

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

cin >> numRows;

RightAngleTriangle triangle(numRows);

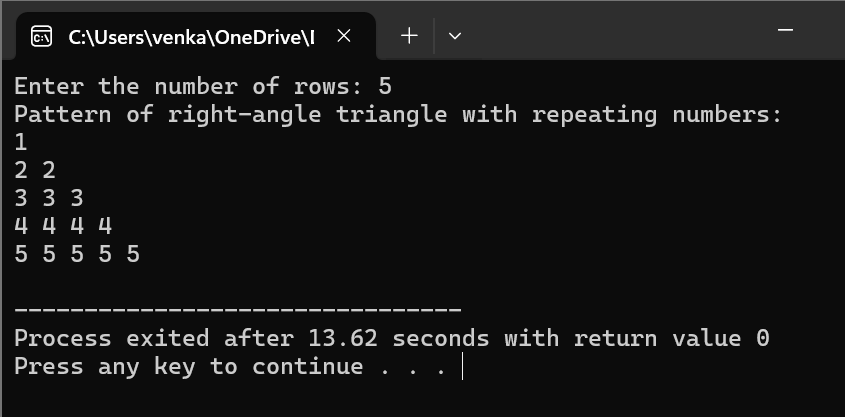
cout << "Pattern of right-angle triangle with repeating numbers:" << endl;

triangle.printPattern();

return 0;

}

**OUTPUT**



**5.PROGRAM**

#include <iostream>

using namespace std;

class ReverseNumber

{

private:

int originalNumber;

int reversedNumber;

public:

ReverseNumber(int num)

{

originalNumber = num;

reversedNumber = reverse(num);

}

int reverse(int num)

{

int reversed = 0;

while (num > 0)

{

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

return reversed;

}

void displayReverse()

{

cout << "Original Number: " << originalNumber << endl;

cout << "Reversed Number: " << reversedNumber << endl;

}

};

int main()

{

int num;

cout << "Enter a number: ";

cin >> num;

ReverseNumber reverseNum(num);

reverseNum.displayReverse();

return 0;

}

OUTPUT

