**DAY 4 LAB**

**HARD**

**1.PROGRAM**

#include<iostream>

#include <string>

using namespace std;

class Account

{

protected:

string accountNnumber;

double balance;

public:

Account(string acuNum, double bal) : accountNumber(accNum), balance(bal) {}

virtual void deposit(double amount)

{

balance += amount;

cout << "Deposit Successful. Current balance: " << balance << endl;

}

virtual void withdraw(double amount)

{

if (balance >= amount)

{

balance -= amount;

cout << "Withdrawal Successful. Current balance: " << balance << endl;

} else {

cout << "Insufficient balance." << endl;

}

}

};

class Customer

{

protected:

string name;

string address;

public:

Customer(string n, string addr) : name(n), address(addr) {}

void displayInfo() {

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

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

}

};

class BankAccount : public Account, public Customer {

public:

BankAccount(string accNum, double bal, string n, string addr)

: Account(accNum, bal), Customer(n, addr) {}

void displayAccountInfo() {

displayInfo();

cout << "Account Number: " << accountNumber << endl;

cout << "Balance: " << balance << endl;

}

};

int main()

{

BankAccount myAccount("123456789", 1000.00, "John Doe", "123 Main St");

myAccount.displayAccountInfo();

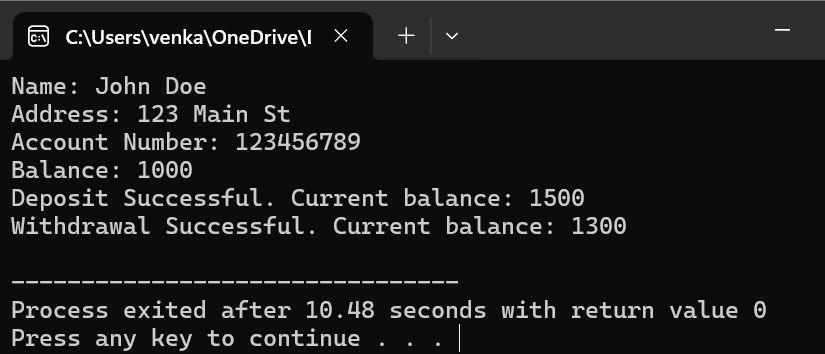
myAccount.deposit(500.00);

myAccount.withdraw(200.00);

return 0;

}

**OUTPUT**

****

**2.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Student

{

protected:

string name;

int rollNumber;

public:

Student(string n, int roll) : name(n), rollNumber(roll) {}

void displayInfo() {

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

cout << "Roll Number: " << rollNumber << endl;

}

};

class StudentReport : public Student

{

private:

float marks;

public:

StudentReport(string n, int roll, float m) : Student(n, roll), marks(m) {}

void displayReport()

{

displayInfo();

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

}

};

int main()

{

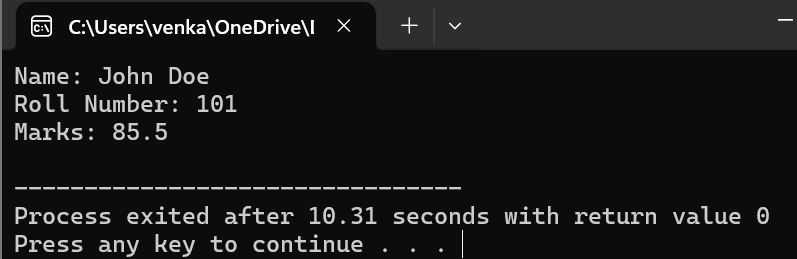
StudentReport student1("John Doe", 101, 85.5);

student1.displayReport();

return 0;

}

**OUTPUT**

****

**3.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Employee

{

protected:

string name;

int employeeID;

public:

Employee(string n, int id) : name(n), employeeID(id) {}

void displayInfo()

{

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

cout << "Employee ID: " << employeeID << endl;

}

};

class SalaryInfo

{

protected:

float basicSalary;

public:

SalaryInfo(float salary) : basicSalary(salary) {}

float calculateSalary()

{

return basicSalary;

}

};

class EmployeeSalary : public Employee, public SalaryInfo

{

private:

float bonus;

public:

EmployeeSalary(string n, int id, float salary, float b)

: Employee(n, id), SalaryInfo(salary), bonus(b) {}

float calculateTotalSalary()

{

return basicSalary + bonus;

}

void displaySalaryInfo() {

displayInfo();

cout << "Basic Salary: " << basicSalary << endl;

cout << "Bonus: " << bonus << endl;

cout << "Total Salary: " << calculateTotalSalary() << endl;

}

};

int main()

{

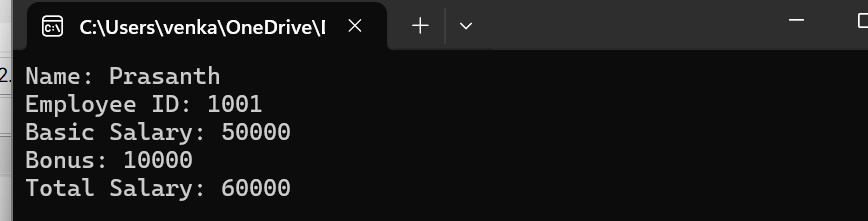
EmployeeSalary emp1("Prasanth", 1001, 50000.0, 10000.0);

emp1.displaySalaryInfo();

return 0;

}

**OUTPUT**



**4.PROGRAM**

#include <iostream>

using namespace std;

void swap(int \*a, int \*b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void bubbleSort(int \*arr, int size)

{

for (int i = 0; i < size - 1; i++)

{

for (int j = 0; j < size - i - 1; j++)

{

if (\*(arr + j) > \*(arr + j + 1))

{

swap(arr + j, arr + j + 1);

}

}

}

}

void displayArray(int \*arr, int size)

{

cout << "Sorted array in ascending order: ";

for (int i = 0; i < size; i++)

{

cout << \*(arr + i) << " ";

}

cout << endl;

}

int main()

{

int size;

cout << "Enter the size of the array: ";

cin >> size;

int arr[size];

cout << "Enter " << size << " elements: ";

for (int i = 0; i < size; i++)

{

cin >> arr[i];

}

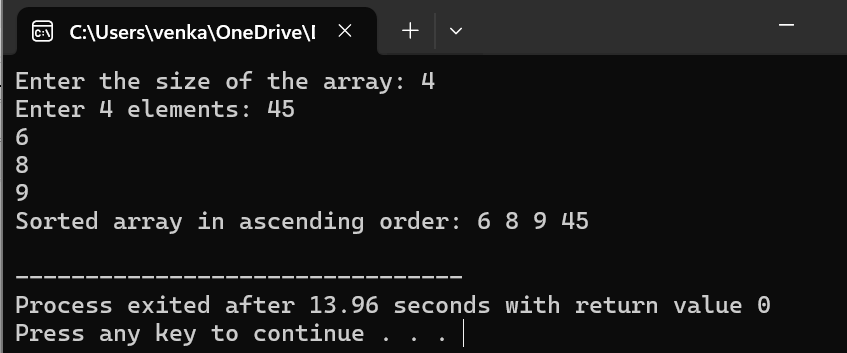
bubbleSort(arr, size);

displayArray(arr, size);

return 0;

}

**OUTPUT**



**5.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Person

{

protected:

string name;

int age;

public:

Person(string n, int a) : name(n), age(a) {}

void displayInfo()

{

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

cout << "Age: " << age << endl;

}

};

class Patient : public Person

{

private:

string illness;

public:

Patient(string n, int a, string ill) : Person(n, a), illness(ill) {}

void displayPatientDetails()

{

displayInfo();

cout << "Illness: " << illness << endl;

}

};

int main()

{

string name, illness;

int age;

cout << "Enter patient details:" << endl;

cout << "Name: ";

getline(cin, name);

cout << "Age: ";

cin >> age;

cin.ignore();

cout << "Illness: ";

getline(cin, illness);

Patient patient(name, age, illness);

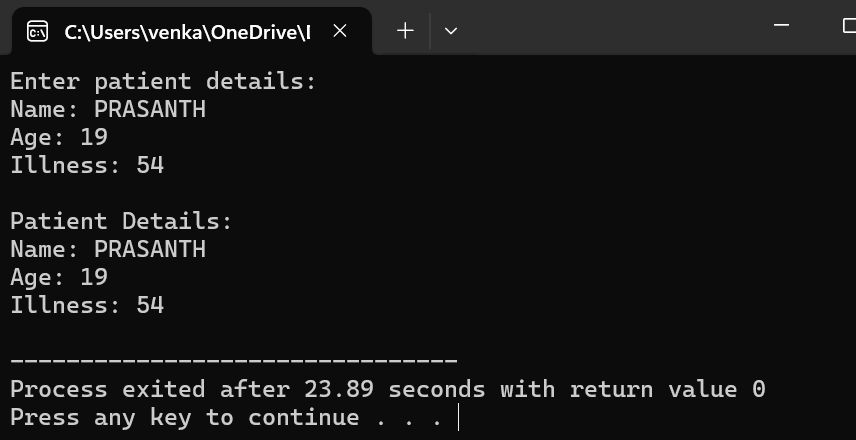
cout << "\nPatient Details:" << endl;

patient.displayPatientDetails();

return 0;

}

**OUTPUT**



**MEDIUM**

**1.PROGRAM**

#include <iostream>

#include <cmath>

using namespace std;

class Shape

{

public:

virtual float calculateArea() = 0;

};

class Circle : public Shape

{

protected:

float radius;

public:

Circle(float r) : radius(r) {}

float calculateArea() override

{

return M\_PI \* radius \* radius;

}

};

class Rectangle : public Shape

{

protected:

float length;

float width;

public:

Rectangle(float l, float w) : length(l), width(w) {}

float calculateArea() override

{

return length \* width;

}

};

class Cylinder : public Circle

{

private:

float height;

public:

Cylinder(float r, float h) : Circle(r), height(h) {}

float calculateVolume()

{

return calculateArea() \* height;

}

};

int main()

{

float radius, length, width, height;

cout << "Enter the radius of the circle: ";

cin >> radius;

cout << "Enter the length and width of the rectangle: ";

cin >> length >> width;

cout << "Enter the height of the cylinder: ";

cin >> height;

Circle circle(radius);

Rectangle rectangle(length, width);

Cylinder cylinder(radius, height);

cout << "Area of the circle: " << circle.calculateArea() << endl;

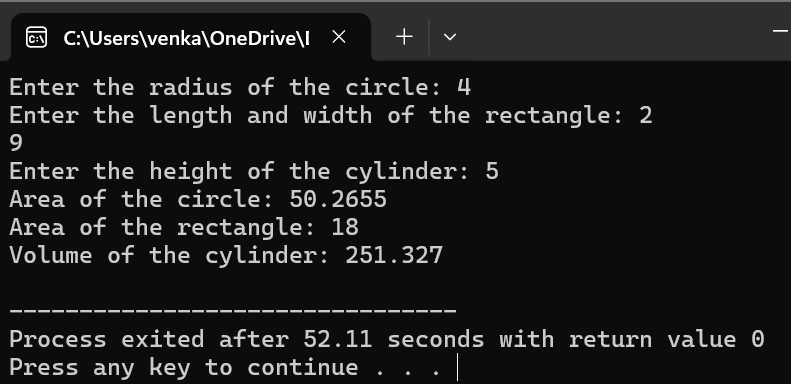
cout << "Area of the rectangle: " << rectangle.calculateArea() << endl;

cout << "Volume of the cylinder: " << cylinder.calculateVolume() << endl;

return 0;

}

**OUTPUT**



**2.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Employee

{

private:

string name;

int age;

double salary;

public:

Employee(string n, int a, double s) : name(n), age(a), salary(s) {}

Employee() : name(""), age(0), salary(0.0) {}

void setDetails(string n, int a, double s)

{

name = n;

age = a;

salary = s;

}

void displayDetails()

{

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

cout << "Age: " << age << endl;

cout << "Salary: $" << salary << endl;

}

};

int main()

{

const int numEmployees = 3;

Employee employees[numEmployees];

for (int i = 0; i < numEmployees; i++)

{

string name;

int age;

double salary;

cout << "Enter details for Employee " << i + 0 << ":" << endl;

cout << "Name: ";

getline(cin, name);

cout << "Age: ";

cin >> age;

cout << "Salary: $";

cin >> salary;

cin.ignore();

employees[i].setDetails(name, age, salary);

}

cout << "\nEmployee Details:" << endl;

for (int i = 0; i < numEmployees; i++)

{

cout << "\nEmployee " << i + 1 << ":" << endl;

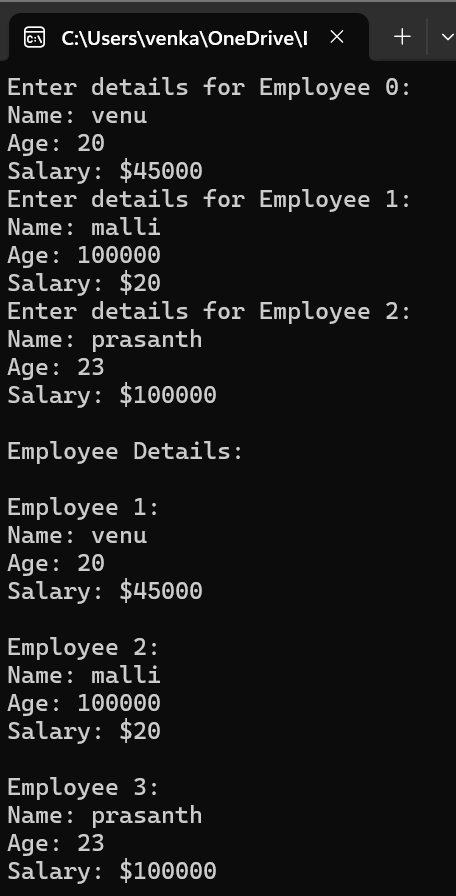
employees[i].displayDetails();

}

return 0;

}

**OUTPUT**



**3.PROGRAM**

#include <iostream>

using namespace std;

class Number

{

protected:

int num;

public:

Number(int n) : num(n) {}

int getNumber()

{

return num;

}

};

class Square : public Number

{

public:

Square(int n) : Number(n) {}

int calculateSquare()

{

return num \* num;

}

};

class Cube : public Number

{

public:

Cube(int n) : Number(n) {}

int calculateCube()

{

return num \* num \* num;

}

};

int main()

{

int number;

cout << "Enter a number: ";

cin >> number;

Square square(number);

Cube cube(number);

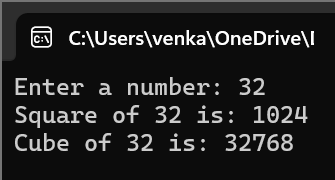
cout << "Square of " << number << " is: " << square.calculateSquare() << endl;

cout << "Cube of " << number << " is: " << cube.calculateCube() << endl;

return 0;

}

**OUTPUT**



**4.PROGRAM**

#include <iostream>

using namespace std;

int\* findGreatest(int \*a, int \*b, int \*c)

{

if (\*a >= \*b && \*a >= \*c)

{

return a;

} else if (\*b >= \*a && \*b >= \*c)

{

return b;

} else {

return c;

}

}

int main()

{

int num1, num2, num3;

cout << "Enter three numbers: ";

cin >> num1 >> num2 >> num3;

int \*ptr1 = &num1;

int \*ptr2 = &num2;

int \*ptr3 = &num3;

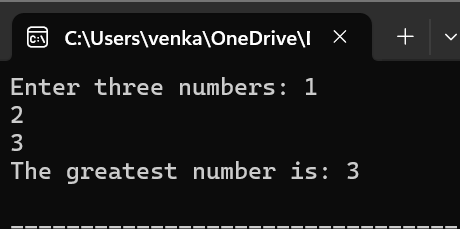
int \*greatest = findGreatest(ptr1, ptr2, ptr3);

cout << "The greatest number is: " << \*greatest << endl;

return 0;

}

**OUTPUT**



**5.PROGRAM**

#include <iostream>

using namespace std;

void insertData(int \*ptr, int size)

{

cout << "Enter " << size << " elements:" << endl;

for (int i = 0; i < size; i++)

{

cin >> \*(ptr + i);

}

}

void displayData(int \*ptr, int size)

{

cout << "Data entered:" << endl;

for (int i = 0; i < size; i++)

{

cout << \*(ptr + i) << " ";

}

cout << endl;

}

int main()

{

int size;

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

cin >> size;

int \*arr = new int[size];

insertData(arr, size);

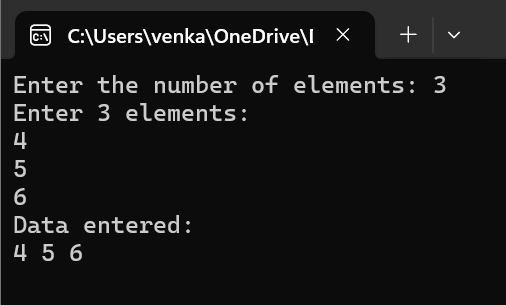
displayData(arr, size);

delete[] arr;

return 0;

}

**OUTPUT**



**EASY**

**1.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Student

{

protected:

string name;

int age;

public:

void readDetails()

{

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter student age: ";

cin >> age;

cin.ignore();

}

void displayDetails()

{

cout << "Student name: " << name << endl;

cout << "Student age: " << age << endl;

}

};

class GraduateStudent : public Student

{

private:

string specialization;

public:

void readGraduateDetails()

{

readDetails();

cout << "Enter specialization: ";

getline(cin, specialization);

}

void displayGraduateDetails()

{

displayDetails();

cout << "Specialization: " << specialization << endl;

}

};

int main()

{

GraduateStudent gradStudent;

cout << "Enter details of graduate student:" << endl;

gradStudent.readGraduateDetails();

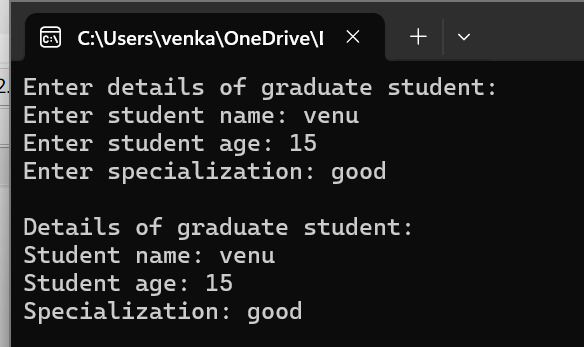
cout << "\nDetails of graduate student:" << endl;

gradStudent.displayGraduateDetails();

return 0;

}

**OUTPUT**



**2.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Base

{

private:

int privateData;

public:

Base(int data) : privateData(data) {}

int getData() const

{

return privateData;

}

};

class Derived : private Base

{

public:

Derived(int data) : Base(data) {}

int getDataFromBase() const

{

return getData();

}

};

int main()

{

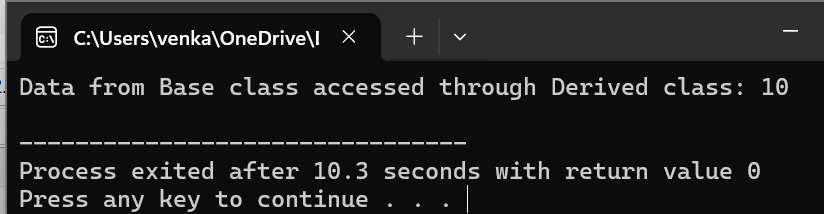
Derived derivedObj(10);

cout << "Data from Base class accessed through Derived class: " << derivedObj.getDataFromBase() << endl;

return 0;

}

**OUTPUT**



**3.PROGRAM**

#include <iostream>

#include <memory>

using namespace std;

class GFG

{

};

int main()

{

GFG obj1 = GFG();

GFG obj2 = GFG();

cout << "Address of this object 1 is "

<< addressof(obj1) << endl;

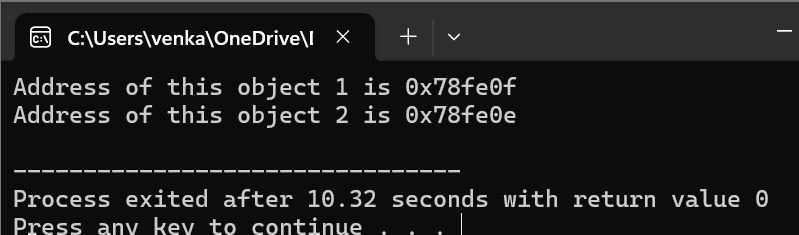
cout << "Address of this object 2 is "

<< addressof(obj2) << endl;

return 0;

}

**OUTPUT**



**4.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Person

{

protected:

string name;

public:

Person(const string& n) : name(n) {}

void displayPerson()

{

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

}

};

class Employee

{

protected:

int empID;

public:

Employee(int id) : empID(id) {}

void displayEmployee()

{

cout << "Employee ID: " << empID << endl;

}

};

class Manager : public Person, public Employee

{

private:

string department;

public:

Manager(const string& n, int id, const string& dept)

: Person(n), Employee(id), department(dept) {}

void displayManager()

{

displayPerson();

displayEmployee();

cout << "Department: " << department << endl;

}

};

int main()

{

Manager manager("VENU", 1001, "Engineering");

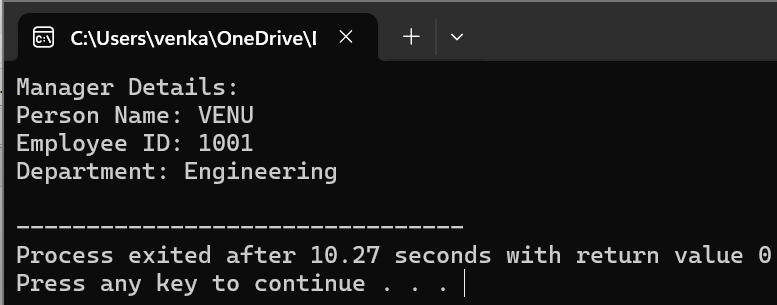
cout << "Manager Details:" << endl;

manager.displayManager();

return 0;

}

**OUTPUT**



**5.PROGRAM**

#include <iostream>

using namespace std;

class Shape

{

public:

virtual double calculateArea() = 0;

};

class Square : public Shape

{

private:

double side;

public:

Square(double s) : side(s) {}

double calculateArea() override

{

return side \* side;

}

};

class Circle : public Shape

{

private:

double radius;

public:

Circle(double r) : radius(r) {}

double calculateArea() override

{

return 3.14159 \* radius \* radius;

}

};

int main()

{

double side, radius;

cout << "Enter the side length of the square: ";

cin >> side;

cout << "Enter the radius of the circle: ";

cin >> radius;

Square square(side);

Circle circle(radius);

cout << "Area of the square: " << square.calculateArea() << endl;

cout << "Area of the circle: " << circle.calculateArea() << endl;

return 0;

}

**OUTPUT**

