

**NAME:** SOURADEEP BHATTACHARYA

**ID:** 201001001035

**BATCH:** BCS2A

**GROUP:** A

**YEAR:** 2nd

**SUBJECT:** Object Oriented

Programming Laboratory

**Index**

| Sl No. | Topic | Date | Remarks |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Assignment - 1**

Problem Statement :

**1.** WAP in C++ to print the following pattern for n lines, where n is taken as

input from the user.

\* 1

\* \* 2

\* \* \* 3

\* \* \* \* 4

Program :

#include <iostream>

using namespace std;

int main(){

int n;

cout<<"Enter the Number of Lines of Pattern to be displayed : ";

cin>>n;

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

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

cout<<" \* ";

}

cout<<i<<endl;

}

return 0;

}

Output :



Problem Statement :

**2.** WAP in C++ to find the odd factors of a number.

Program :

#include <iostream>

using namespace std;

int main(){

int num,fact;

cout<<"Enter a Number : ";

cin>>num;

cout << "Odd Factors of " << num << " are: ";

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

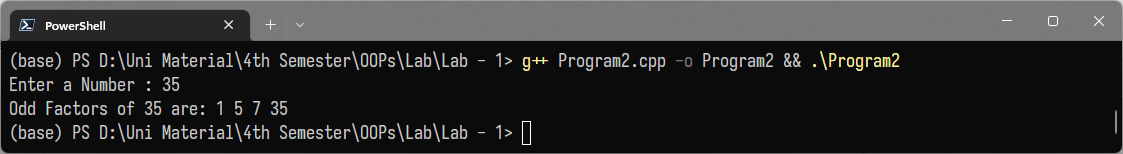
if(num % i == 0 && i%2 != 0)

cout << i << " ";

}

}

Output :



Problem Statement :

**3.** WAP in C++ using functions to check if

(a) a number is divisible by 5 and 7

(b) the product of digits is equal to the sum of digits of a number

Program :

#include <iostream>

#include <string>

#include <Math.h>

using namespace std;

bool divByFiveSeven(int);

bool prodEqualsSum(int);

int main(){

int num;

cout << "Enter a Number : ";

cin >> num;

string result1 = divByFiveSeven(num) ? " Divisible by 5 and 7.\n" : " Not Divisible by 5 and 7.\n";

cout << "The Number is :" << result1;

string result2 = prodEqualsSum(num) ? "Product of digits = Sum of digits \n" : "Product of digits != Sum of digits \n";

cout << result2;

return 0;

}

bool divByFiveSeven(int num){

if (num % 5 == 0 && num % 7 == 0){

return true;

}

return false;

}

bool prodEqualsSum(int num){

int dig = (int)floor(log10(num)) + 1;

int prod = 1, sum = 0;

for (; num > 0; num /= 10){

sum += num % 10;

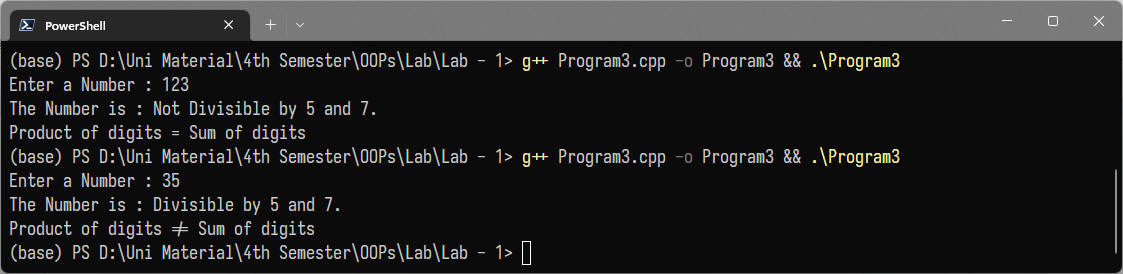
prod \*= num % 10;

}

return (sum == prod) ? true : false;

}

Output :



Problem Statement :

**4.** Implement a structure for time

struct Time {

int hr;

int min;

int sec;

};

Take input from the user. Convert into seconds and print.

Program :

#include <iostream>

using namespace std;

struct time{

int hr;

int min;

int sec;

};

int main(){

struct time t1;

long int seconds = 0;

cout<<"Enter hours : ";

cin>>t1.hr;

cout<<"Enter minutes : ";

cin>>t1.min;

cout<<"Enter seconds : ";

cin>>t1.sec;

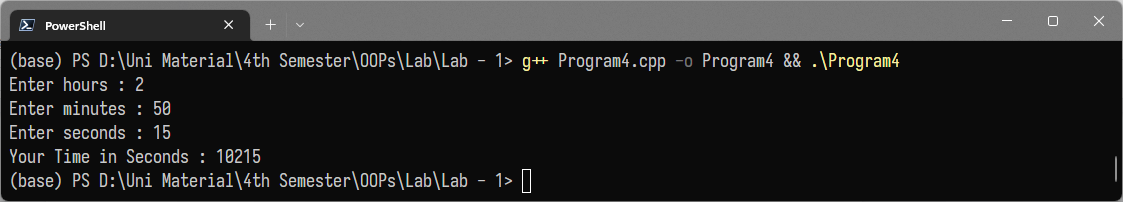
seconds += (t1.hr\*60\*60) + (t1.min\*60) + (t1.sec);

cout<<"Your Time in Seconds : "<<seconds;

return 0;

}

Output :



**Assignment - 2**

Problem Statement :

**1.** (a) create a class Student with public data member name and private

data members id and marks.

(b) write a public set and get methods for the private data members.

Program :

#include <iostream>

#include <string.h>

#include <stdio.h>

using namespace std;

class Student {

private:

int id;

int marks;

public:

char name[50];

void setvalues(int a, int b, char \*n);

void getvalues();

};

void Student::setvalues(int a, int b, char \*n){

strcpy(name, n);

id = a;

marks = b;

}

void Student::getvalues() {

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

cout << "ID is:" << id << endl;

cout << "MARKS is:" << marks << endl;

}

int main() {

Student S;

int identity, marks\_fetched;

char nme[50];

cout << "Enter the name :";

cin.getline(nme, 50);

cout << "Enter the id of the student :";

cin >> identity;

cout << "Enter the marks of the student :";

cin >> marks\_fetched;

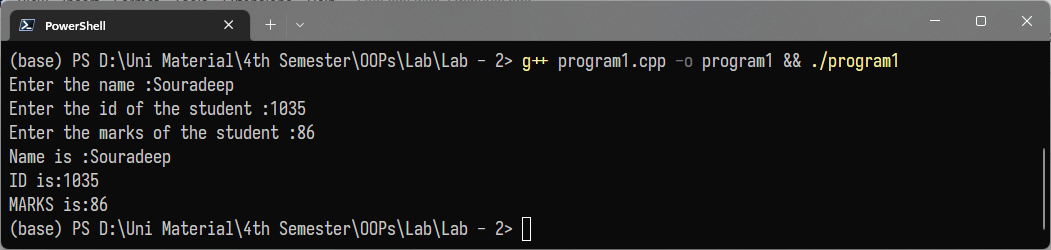
S.setvalues(identity, marks\_fetched, nme);

S.getvalues();

return 0;

}

Output :



Problem Statement :

**2.** Implement a structure for distance

struct Distance {

int feet;

int inches;

};

Take two objects of Distance as length and width of a room. Find the area

of the room in sq feet.

Program :

#include <iostream>

using namespace std;

struct Distance {

int feet;

int inches;

};

struct Room {

Distance length;

Distance breadth;

};

int main() {

struct Distance D;

struct Room R;

cout << "Enter the feet :";

cin >> R.length.feet;

cout << "Enter the inches :";

cin >> R.length.inches;

cout << "Enter the feet :";

cin >> R.breadth.feet;

cout << "Enter the inches :";

cin >> R.breadth.inches;

double l = R.length.feet + R.length.inches / 12.0;

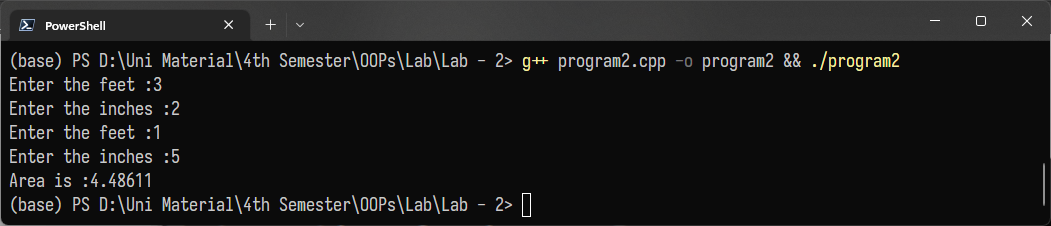
double b = R.breadth.feet + R.breadth.inches / 12.0;

double area;

area = l \* b;

cout << "Area is :" << area;

}

Output :   


Problem Statement :

**3.** (a) in the above program use a constructor to initialize id to

201001001000 and marks to 0.

(b) take the last 3 digits from the user and construct the full id. Take the

name and marks as input. Display name,full id and marks of two

students.

Program :

#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

class Student {

private:

long long int id;

int marks;

public:

Student();

void setid(int num);

char name[50];

void setvalues(int b, char \*n);

void getvalues();

};

Student::Student() {

id = 201001001000;

marks = 0;

}

void Student::setid(int num) {

id += num;

// marks=0;

}

void Student::setvalues(int b, char \*n) {

strcpy(name, n);

marks = b;

}

void Student::getvalues() {

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

cout << "ID is:" << id << endl;

cout << "MARKS is:" << marks << endl;

}

int main() {

Student S;

int marks\_fetched;

long n1;

int s;

char nme[50];

cout << "Enter the number of students you want :";

cin >> s;

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

{

cout << "Enter the name of " << i << "student :";

cin >> nme;

cout << "Enter the marks of " << i << "student :";

cin >> marks\_fetched;

cout << "Enter the last 3 digits of" << i << "student :";

cin >> n1;

S.setid(n1);

S.setvalues(marks\_fetched, nme);

S.getvalues();

}

return 0;

}

Output :



**Assignment – 3**

Problem Statement :

**1.** Suppose you are asked to model a toy-train ride at an amusement park. Passengers are expected to pay a Rs. 50 fee. Children under 5 years of age do not need a ticket. The ticket counter keeps track of the number of passengers that have gone by, and of the total amount of money collected.

Model this ticket-counter with a class called ticketCounter. The two data items are a type unsigned int to hold the total number of passengers, and a type double to hold the total amount of money collected. A constructor initializes both of these to 0. A member function called payingPassenger() increments the passenger total and adds Rs. 50 to the cash total. Another function called childPassenger(), increments the passenger total, but does not add anything to the cash total. Finally, a member function called display() displays the two totals.

Include a program to test this class. The program should allow a ticket-collector to press one key to count a paying passenger, another key to count a child passenger. By pushing a third key the collector can print the total number of passengers, total amount of cash collected and then exit.

Program :

#include <bits/stdc++.h>

using namespace std;

class ticketCounter

{

unsigned int number\_of\_passengers;

double money\_collected;

public:

ticketCounter(){

number\_of\_passengers = 0;

money\_collected = 0;

}

void payingPassenger(){

number\_of\_passengers += 1;

money\_collected += 50;

}

void childPassenger() {

number\_of\_passengers += 1;

}

void display() {

cout << "NUMBER OF PASSENGERS : " << number\_of\_passengers << endl;

cout << "MONEY COLLECTED : " << money\_collected << endl;

}

};

int main() {

ticketCounter tc;

cout << "Welcome to Ticket Counter\nPress 1 to add a Paying Passenger" << endl;

cout << "Press 2 to add a Child Passenger\nPress 0 to view current Passenger Count and Money Collected and Exit"<<endl;

int ch = 1;

while (ch) {

cin >> ch;

switch (ch) {

case 0:

tc.display();

break;

case 1:

tc.payingPassenger();

break;

case 2:

tc.childPassenger();

break;

default:

cout << "WRONG INPUT.TRY AGAIN" << endl;

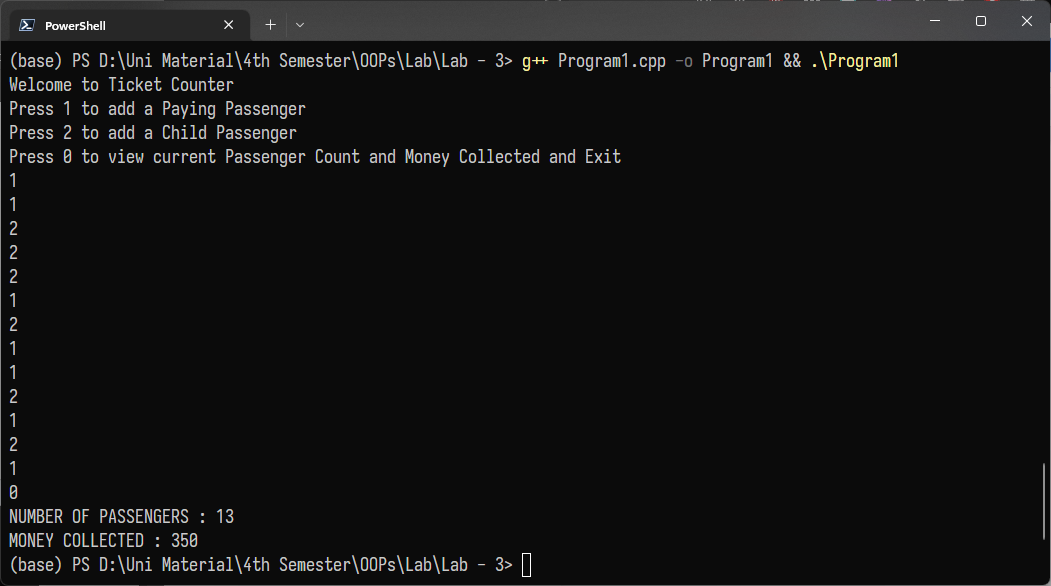
}

}

return 0;

}

Output :



Problem Statement :

**2.** Create a linked list using a class for a node. Use new to create a node object and delete to destroy it.Implement the following operations:

(a) Insert at beginning and end.

(b) Delete from any location.

Program :

#include <iostream>

#include <stdlib.h>

using namespace std;

class node

{

public:

int v;

node \*next;

node()

{

next = NULL;

}

};

class LinkedList

{

node \*head;

public:

LinkedList();

void insert\_at\_beginning(int);

void insert\_at\_end(int);

void delete\_at\_given\_position(int);

void print();

};

LinkedList::LinkedList()

{

head = NULL;

}

void LinkedList::insert\_at\_beginning(int v)

{

node \*temp = new node();

temp->v = v;

temp->next = head;

head = temp;

}

void LinkedList::insert\_at\_end(int v)

{

node \*temp = new node();

temp->v = v;

if (head == NULL)

{

head = temp;

}

else

{

node \*ptr = head;

while (ptr->next != NULL)

{

ptr = ptr->next;

}

ptr->next = temp;

}

}

void LinkedList::delete\_at\_given\_position(int p)

{

if (head == NULL)

{

cout << "List is Empty" << endl;

}

else

{

node \*temp, \*ptr;

if (p == 0)

{

cout << "Element Deleted: " << head->v << endl;

ptr = head;

head = head->next;

delete (ptr);

}

else

{

temp = ptr = head;

while (p > 0)

{

--p;

temp = ptr;

ptr = ptr->next;

}

cout << "Element Deleted: " << ptr->v << endl;

temp->next = ptr->next;

free(ptr);

}

}

}

void LinkedList::print()

{

if (head == NULL)

{

cout << "List is empty" << endl;

}

else

{

node \*temp = head;

cout << "Linked List: ";

while (temp != NULL)

{

cout << temp->v << "->";

temp = temp->next;

}

cout << "NULL" << endl;

}

}

int main()

{

int choice, v, p;

LinkedList ll;

while (1)

{

printf("1 to Insert at the beginning");

printf("\n2 to Insert at the end");

printf("\n3 to Delete from at any given position");

printf("\n4 to Display");

printf("\n5 to Exit");

cout << "\nEnter Your Choice: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Enter Element: ";

cin >> v;

ll.insert\_at\_beginning(v);

break;

case 2:

cout << "Enter Element: ";

cin >> v;

ll.insert\_at\_end(v);

break;

case 3:

cout << "Enter Position : ";

cin >> p;

ll.delete\_at\_given\_position(p - 1);

break;

case 4:

ll.print();

break;

case 5:

exit(0);

break;

default:

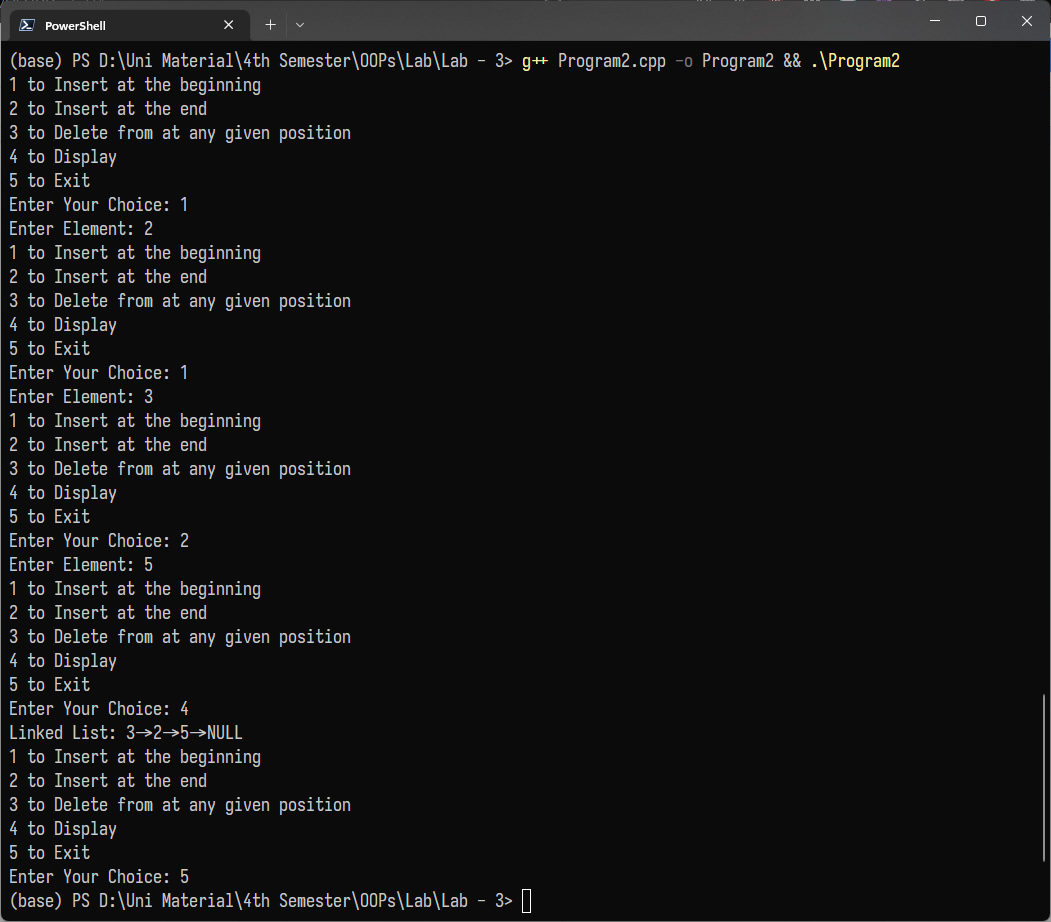
cout << "Wrong choice!!";

}

}

}

Output :



Problem Statement :

**3.** Update the linked list implementation to work as a stack and implement push and pop operations.

Program :

#include <iostream>

using namespace std;

class Node

{

public:

int data;

Node \*next;

Node()

{

data = 0;

next = NULL;

}

};

class llist

{

Node \*head;

public:

llist();

void push(int);

void printList();

void pop();

};

llist::llist()

{

head = NULL;

}

void llist::push(int data)

{

Node \*newNode = new Node();

newNode->data = data;

if (head == NULL)

{

head = newNode;

return;

}

Node \*temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = newNode;

}

void llist::pop()

{

Node \*temp1 = head, \*temp2 = NULL;

if (head == NULL)

{

cout << "Underflow" << endl;

return;

}

if (head->next == NULL)

{

head = NULL;

return;

}

temp1 = head;

while (temp1->next != NULL)

{

temp2 = temp1;

temp1 = temp1->next;

}

temp2->next = NULL;

delete temp1;

}

void llist::printList()

{

Node \*temp = head;

if (head == NULL)

{

cout << "List empty" << endl;

return;

}

while (temp != NULL)

{

cout << temp->data << " ";

temp = temp->next;

}

}

int main()

{

llist list;

int ch, n, size = 0;

while (1)

{

cout << "\nEnter your choice -\n 1 to push\n 2 to pop \n 3 to display\n \n0 to

exit\n-- -

";

cin >>

ch;

switch (ch)

{

case 1:

cout << "\nEnter a number ";

cin >> n;

list.push(n);

size++;

break;

case 2:

list.pop();

break;

case 3:

list.printList();

break;

case 0:

return 1;

break;

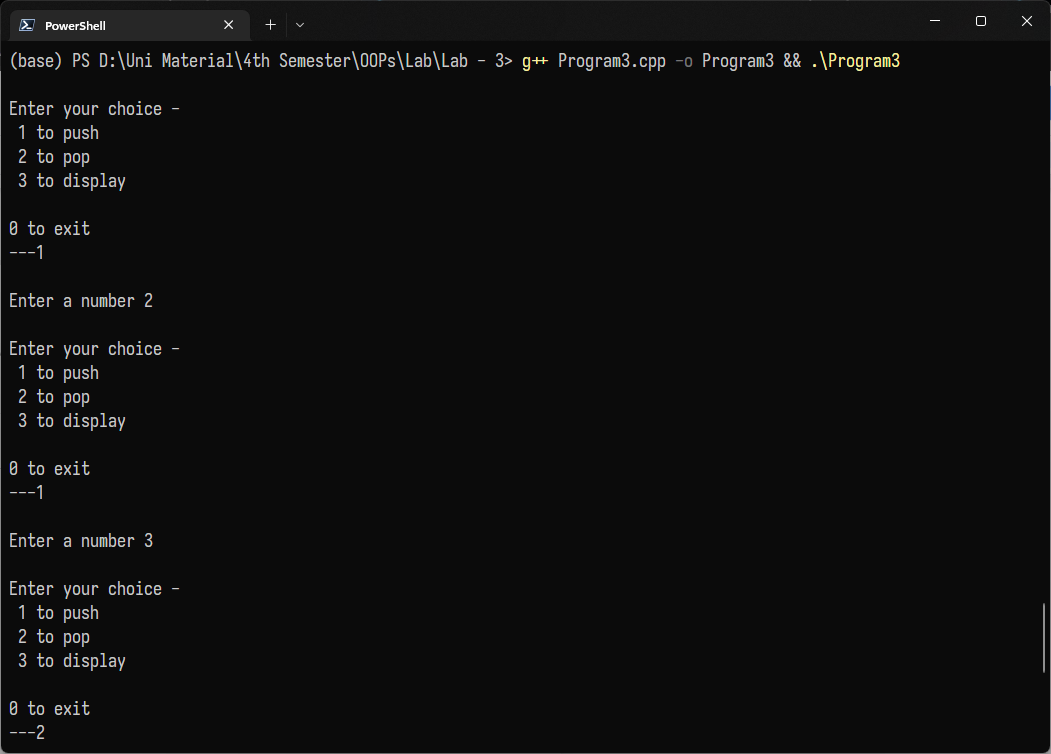
}

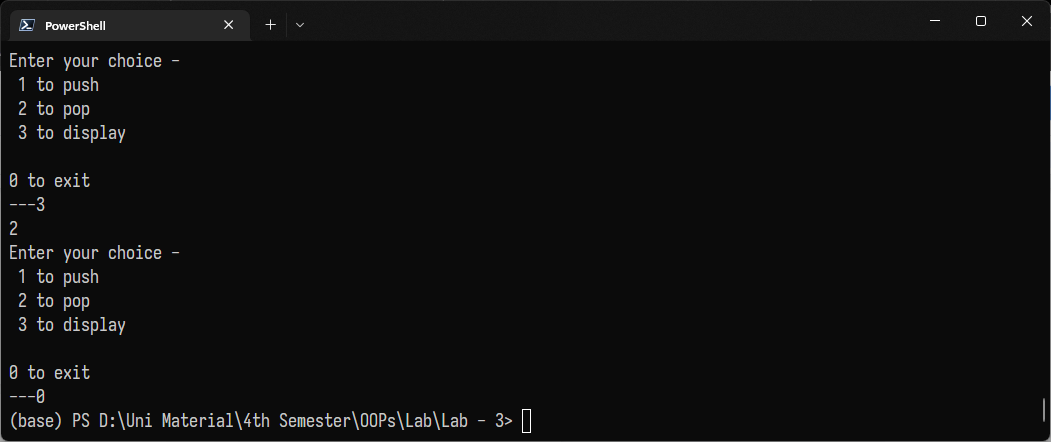
}

return 0;

}

Output :





**Assignment – 4**

Problem Statement :

**1.**An election is contested by 5 candidates. The candidates are numbered 1

to 5 and the voting is done by marking the candidate number in the ballot

paper. Create a class BallotBox to read the ballots and count the votes cast

for each candidate using an array variable count. In case a number read is

outside the range 1 to 5 the ballot should be considered as spoilt ballot

and the program should also count the number of spoiled ballots.

Program :

#include <iostream>

using namespace std;

class BallotBox

{

int votes;

public:

void read(int v)

{

votes = v;

}

int count[6] = {0, 0, 0, 0, 0, 0};

void counter(){

if (votes == 1){

count[0]++;

}

else if (votes == 2){

count[1]++;

}

else if (votes == 3){

count[2]++;

}

else if (votes == 4){

count[3]++;

}

else if (votes == 5){

count[4]++;

}

else{

count[5]++;

}

}

};

int main(){

BallotBox Box;

int v, n;

cout << "enter number of voters";

cin >> n;

cout << "\n\tCANDIDATES ARE NUMBERED FROM 1 TO 5" << endl;

cout << "\tCAST YOUR VOTES :-" << endl;

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

{

cin >> v;

Box.read(v);

Box.counter();

}

cout << "\n\tRESULTS :-" << endl;

cout << "Candidate 1 : " << Box.count[0] << endl;

cout << "Candidate 2 : " << Box.count[1] << endl;

cout << "Candidate 3 : " << Box.count[2] << endl;

cout << "Candidate 4 : " << Box.count[3] << endl;

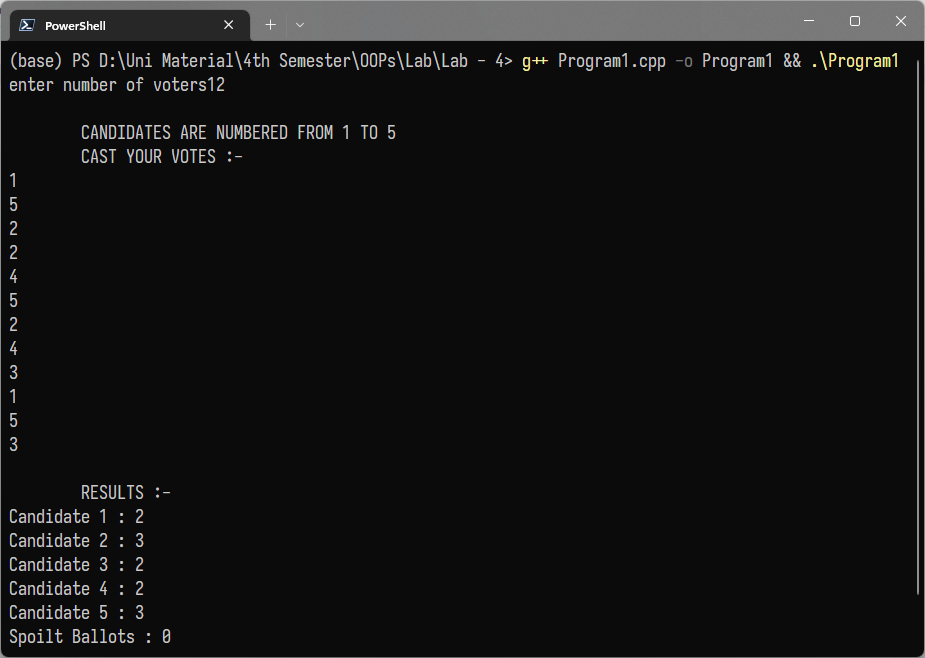
cout << "Candidate 5 : " << Box.count[4] << endl;

cout << "Spoilt Ballots : " << Box.count[5] << endl;

return 0;

}

Output :



Problem Statement :

**2.**Write a C++ program using array of objects that store player name, matches

played and runs scored for a given number of players. Involve member functions to obtain player with maximum matches played, maximum runs scored

and maximum average.

Program :

#include <bits/stdc++.h>

using namespace std;

class Player

{

string name;

int match\_played;

int runs\_scored;

public:

void set\_credendtials(string s, int mp, int rs)

{

name = s;

match\_played = mp;

runs\_scored = rs;

}

Player maximum\_matches\_played(Player player[], int n)

{

int max\_player, max\_match = INT\_MIN;

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

if (max\_match < player[i].match\_played)

{

max\_player = i;

max\_match = player[i].match\_played;

}

return player[max\_player];

}

Player maximum\_run\_scored(Player player[], int n)

{

int max\_player, max\_run = INT\_MIN;

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

if (max\_run < player[i].runs\_scored)

{

max\_player = i;

max\_run = player[i].runs\_scored;

}

return player[max\_player];

}

Player maximum\_average(Player player[], int n)

{

double max\_average = 0.0;

int max\_player;

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

if (max\_average < player[i].runs\_scored / (player[i].match\_played \* 1.00))

{

max\_player = i;

max\_average = player[i].runs\_scored / (player[i].match\_played \* 1.00);

}

return player[max\_player];

}

void display()

{

cout << name << endl;

}

};

int main()

{

int number\_of\_players;

string name;

int mp, rs;

cout << "Input the number of players : ";

cin >> number\_of\_players;

Player

player\_list[number\_of\_players],

player\_with\_maximum\_matches\_played, player\_with\_maximum\_runs\_scored, player\_with\_maximum\_average;

for (int i = 0; i < number\_of\_players; i++)

{

cout << "Input the details for player " << i + 1 << " : " << endl;

cout << "Name : ";

cin >> name;

cout << "Number of matches played : ";

cin >> mp;

cout << "Total runs scored : ";

cin >> rs;

player\_list[i].set\_credendtials(name, mp, rs);

}

player\_with\_maximum\_matches\_played = player\_with\_maximum\_matches\_played.maximum\_matches\_played(player\_list,number\_of\_players);

player\_with\_maximum\_runs\_scored = player\_with\_maximum\_runs\_scored.maximum\_run\_scored(player\_list, number\_of\_players);

player\_with\_maximum\_average = player\_with\_maximum\_average.maximum\_average(player\_list, number\_of\_players);

cout << "Player who has played the maximum number of matches is :";player\_with\_maximum\_matches\_played.display();

cout<< "Player who has scored the maximum runs is : ";

player\_with\_maximum\_runs\_scored.display();

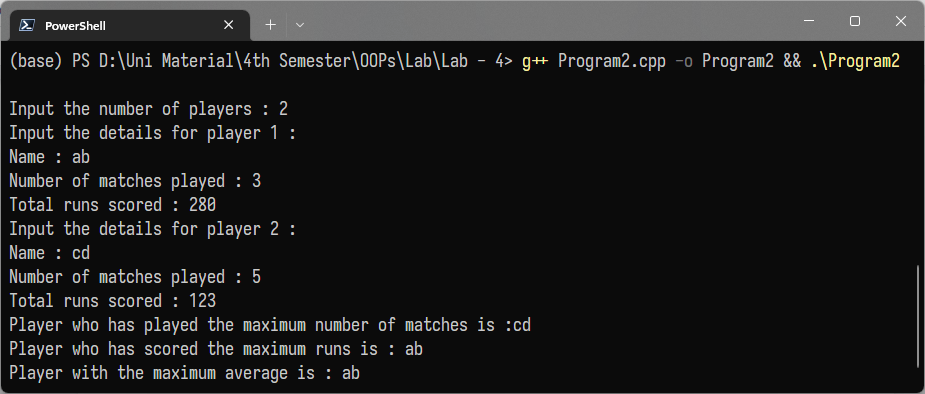
cout << "Player with the maximum average is : ";

player\_with\_maximum\_average.display();

return 0;

}

Output :



Problem Statement :

**3.** Write a function called reversit() that reverses a C-string (an array of

char). Use a for loop that swaps the first and last characters, then the

second and next-to-last characters, and so on. The string should be passed

to reversit() as an argument. Write a program to exercise reversit().

The program should get a string from the user, call reversit(), and print

out the result. Use an input method that allows embedded blanks.

Program :

#include <iostream>

#include <string>

using namespace std;

string reversitt(string s)

{

char c;

int len = s.length();

int n = len - 1;

for (int i = 0; i < (len / 2); i++)

{

c = s[i];

s[i] = s[n];

s[n] = c;

n = n - 1;

}

return s;

}

int main()

{

string s, s1;

cout << "enter a string" << endl;

cin >> s;

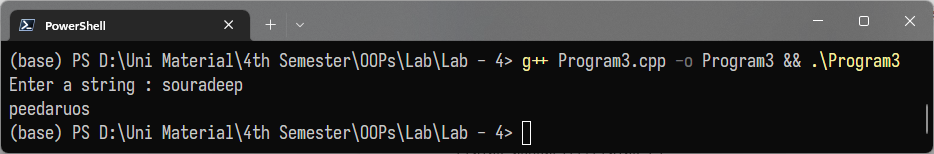
s1 = reversitt(s);

cout << s1 << endl;

return 0;

}

Output :



**Assignment – 5**

Problem Statement :

**1.** Write overloaded functions to add two numbers depending on the types for each of

the following cases :

a)add two int and return int

b)add three int and return int

c)add two float and return float

d)add one int to double and return double

e)add one double to int and return double

Write the main function to execute these functions.

Program :

#include <bits/stdc++.h>

using namespace std;

int add(int a, int b)

{

cout << "calling add(int,int)" << endl;

cout << "Answer : ";

return a + b;

}

int add(int a, int b, int c)

{

cout << "calling add(int,int,int)" << endl;

cout << "Answer : ";

return a + b + c;

}

float add(float a, float b)

{

cout << "calling add(float,float)" << endl;

cout << "Answer : ";

return a + b;

}

double add(int a, double b)

{

cout << "calling add(int,double)" << endl;

cout << "Answer : ";

return a + b;

}

double add(double a, int b)

{

cout << "calling add(double,int)" << endl;

cout << "Answer : ";

return a + b;

}

int main()

{

int a, b, e;

float c, f;

double d;

cout << "Enter number for addition of 2 integers : ";

cin >> a >> b;

cout << add(a, b) << endl;

cout << "Enter number for addition of 3 integers : ";

cin >> a >> b >> e;

cout << add(a, b, e) << endl;

cout << "Enter number for addition of 2 floating point numbers : ";

cin >> c >> f;

cout << add(c, f) << endl;

cout << "Enter number for addition of integer to double number : ";

cin >> a >> d;

cout << add(a, d) << endl;

cout << "Enter number for addition of double number to integer : ";

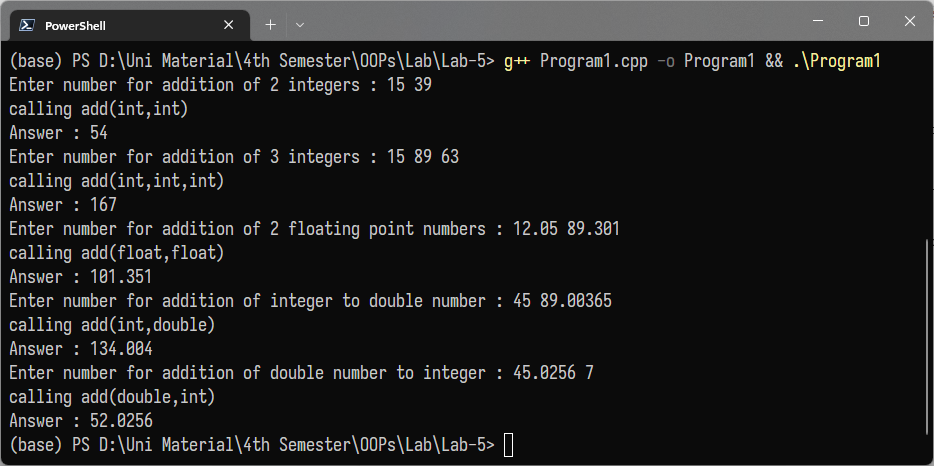
cin >> d >> a;

cout << add(d, a) << endl;

return 0;

}

Output :



Problem Statement :

**2.** In the above program (Program 1) use default parameters to write one

function for a) and b)

Program :

#include <bits/stdc++.h>

using namespace std;

int add(int a, int b, int c = 0) {

cout << "calling add(int,int,int)" << endl;

cout << "Answer : ";

return a + b + c;

}

float add(float a, float b) {

cout << "calling add(float,float)" << endl;

cout << "Answer : ";

return a + b;

}

double add(int a, double b) {

cout << "calling add(int,double)" << endl;

cout << "Answer : ";

return a + b;

}

double add(double a, int b) {

cout << "calling add(double,int)" << endl;

cout << "Answer : ";

return a + b;

}

int main() {

int a, b, e;

float c, f;

double d;

cout << "Enter number for addition of 2 integers : ";

cin >> a >> b;

cout << add(a, b) << endl;

cout << "Enter number for addition of 3 integers : ";

cin >> a >> b >> e;

cout << add(a, b, e) << endl;

cout << "Enter number for addition of 2 floating point numbers : ";

cin >> c >> f;

cout << add(c, f) << endl;

cout << "Enter number for addition of integer to double number : ";

cin >> a >> d;

cout << add(a, d) << endl;

cout << "Enter number for addition of double number to integer : ";

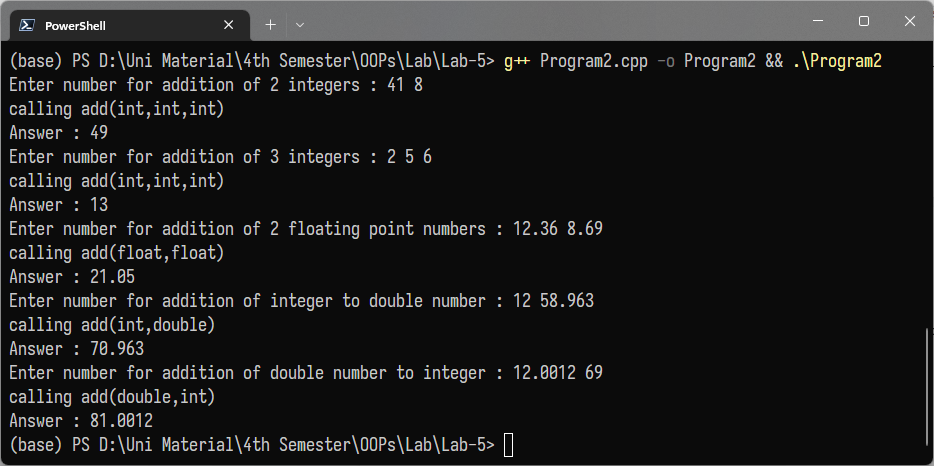
cin >> d >> a;

cout << add(d, a) << endl;

return 0;

}

Output :



Problem Statement :

**3.**Write overloaded functions to calculate area according to the

declaration given below :

a)int area(int side) //calculate the area of a square

b)int area(int length , int breadth) //calculate the area of a rectangle

c)float area(float radius) //calculate the area of a circle

Write a main function to test these functions.

Program :

#include <bits/stdc++.h>

using namespace std;

int area(int side)

{

return side \* side;

}

int area(int length, int breadth)

{

return length \* breadth;

}

float area(float radius)

{

return 3.14579 \* radius \* radius;

}

int main()

{

int side, length, breadth;

float radius;

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

cin >> side;

cout << "The area of the square is : " << area(side) << endl;

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

cin >> length >> breadth;

cout << "The area of the rectangle is : " << area(length, breadth) << endl;

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

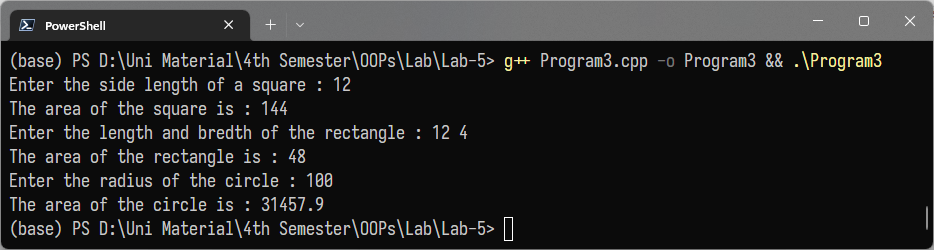
cin >> radius;

cout << "The area of the circle is : " << area(radius) << endl;

return 0;

}

Output :



**Assignment – 6**

Problem Statement :

1. Create a base class employee with:

1. Public data members name and id

From this class derive the other classes using public Derivation:

a. Manager class with

i. Private data member salary.

ii. Public data member designation (e.g., president, vice-

president etc.).

b. Scientist class with

i. Private data member salary.

ii. Public data member num\_of\_publications.

Create one object of manager and one of object for

scientist. Take Input from user for all four data fields for

each object and display

Program :

#include <iostream>

using namespace std;

class employee

{

public:

string name;

int id;

};

class manager : public employee

{

private:

int salary;

public:

char designation[30];

void getdata()

{

cout << "\n\nEnter the name: \n";

cin >> name;

cout << "Enter the id: \n";

cin >> id;

cout << "Enter the designation: \n";

cin >> designation;

cout << "Enter the salary: \n";

cin >> salary;

}

void show()

{

cout << "\nManager Name:"<<name<<"\nId : "<<id<<"\nDesignation : "<<designation<<"\nSalary : "<<salary;

}

};

class scientist : public employee

{

private : int salary;

public:

int num\_of\_publication;

void getdata()

{

cout << "\n\nEnter the name of scientist: \n";

cin >> name;

cout << "Enter the id: \n";

cin >> id;

cout << "Enter the number of publication: \n";

cin >> num\_of\_publication;

cout << "Enter the salary: \n";

cin >> salary;

}

void show()

{

cout << "\nScientist Name: " << name << "\nId:" << id << "\nNumber of publication : "<<num\_of\_publication<<"\nSalary : "<<salary;

}

};

int main()

{

manager m;

m.getdata();

m.show();

scientist s;

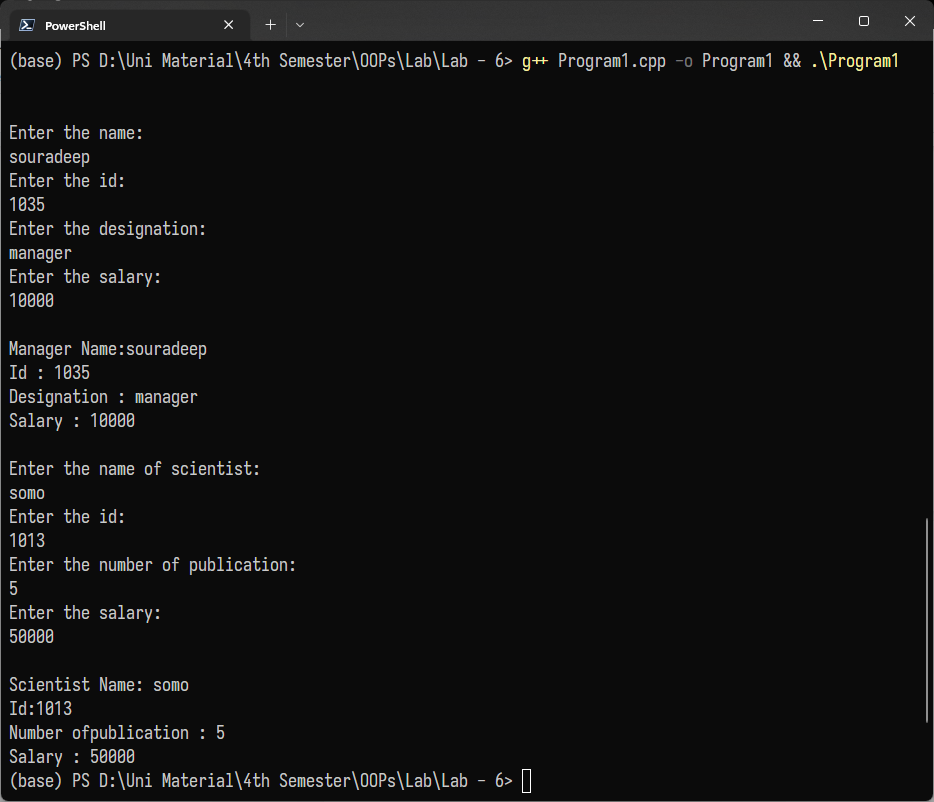
s.getdata();

s.show();

return 0;

}

Output :



Problem Statement :

2. In the above program, use private derivation. What are the

public methods that you need to add?

Program :

#include <iostream>

using namespace std;

class employee

{

public:

int id;

char name[50];

};

class Manager : private employee

{

int salary;

public:

char de[50];

void getData()

{

cout << "\nEnter manager details:\nName :";

cin >> name;

cout << "\nID :";

cin >> id;

cout << "\nDesignation :";

cin >> de;

cout << "\nSalary :";

cin >> salary;

}

void show(int i)

{

cout << "\nEnter manager details:" << i << endl;

cout << "\nNAME-" << name;

cout << "\nID :" << id;

cout << "\nDesignation :" << de;

cout << "\nSalary :" << salary;

}

};

class Scientist : private employee

{

int sal;

public:

int nop;

void getData()

{

cout << "\nEnter Scientist details:\nName :";

cin >> name;

cout << "\nSalary :";

cin >> sal;

cout << "\nnumber of publication :";

cin >> nop;

}

void show(int i)

{

cout << "\nEnter Scientist details:" << i << endl;

cout << "\nNAME-" << name;

cout << "\nID :" << id;

cout << "\nSalary :" << sal;

cout << "\nNumber of publication :" << nop;

}

};

int main()

{

Manager m1, m2;

Scientist s1, s2;

m1.getData();

s2.getData();

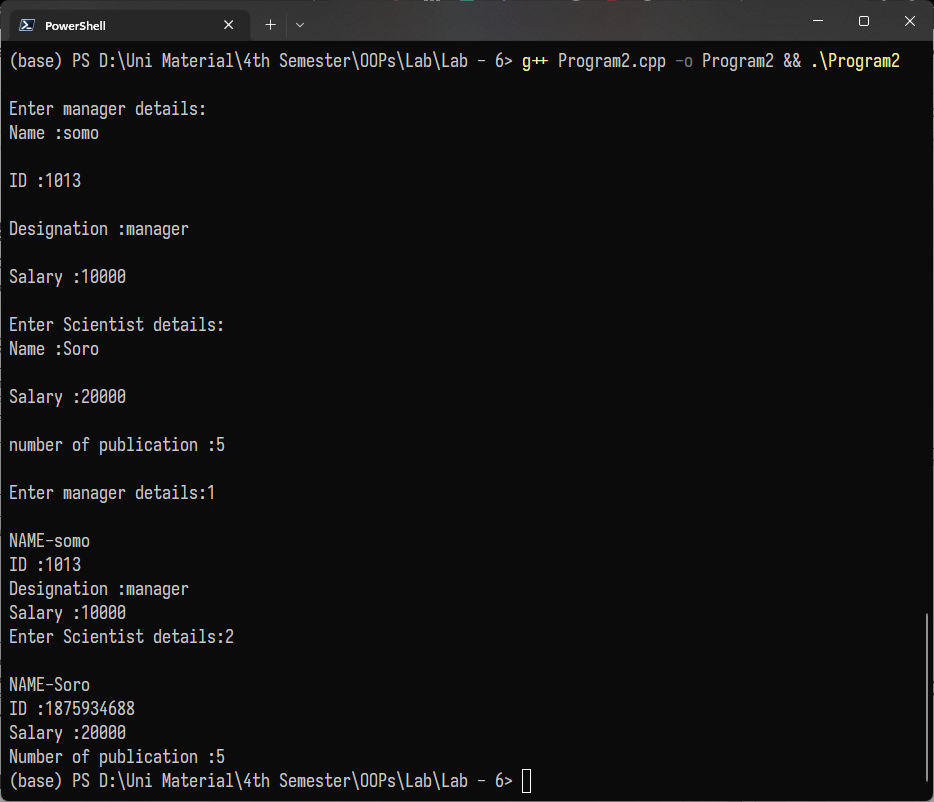
m1.show(1);

s2.show(2);

return 0;

}

Output :



Problem Statement :

3. In the same program, now make the id protected. Add necessary methods to access id in private derivation of the manager and scientist classes.

Program :

#include <iostream>

using namespace std;

class employee

{

protected:

string name;

int id;

};

class manager : public employee

{

private:

int salary;

public:

char designation[30];

void getdata()

{

cout << "\n\nEnter the name: \n";

cin >> name;

cout << "Enter the id: \n";

cin >> id;

cout << "Enter the designation: \n";

cin >> designation;

cout << "Enter the salary: \n";

cin >> salary;

}

void show()

{

cout << "\nManager Name:"<<name<<"\nId : "<<id<<"\nDesignation : "<<designation<<"\nSalary : "<<salary;

}

};

class scientist : public employee

{

private:

int salary;

public:

int num\_of\_publication;

void getdata()

{

cout << "\n\nEnter the name of scientist: \n";

cin >> name;

cout << "Enter the id: \n";

cin >> id;

cout << "Enter the number of publication: \n";

cin >> num\_of\_publication;

cout << "Enter the salary: \n";

cin >> salary;

}

void show()

{

cout << "\nScientist Name: " << name << "\nId:" << id << "\nNumber of publication : "<<num\_of\_publication<<"\nSalary : "<<salary;

}

};

int main()

{

manager m;

m.getdata();

m.show();

scientist s;

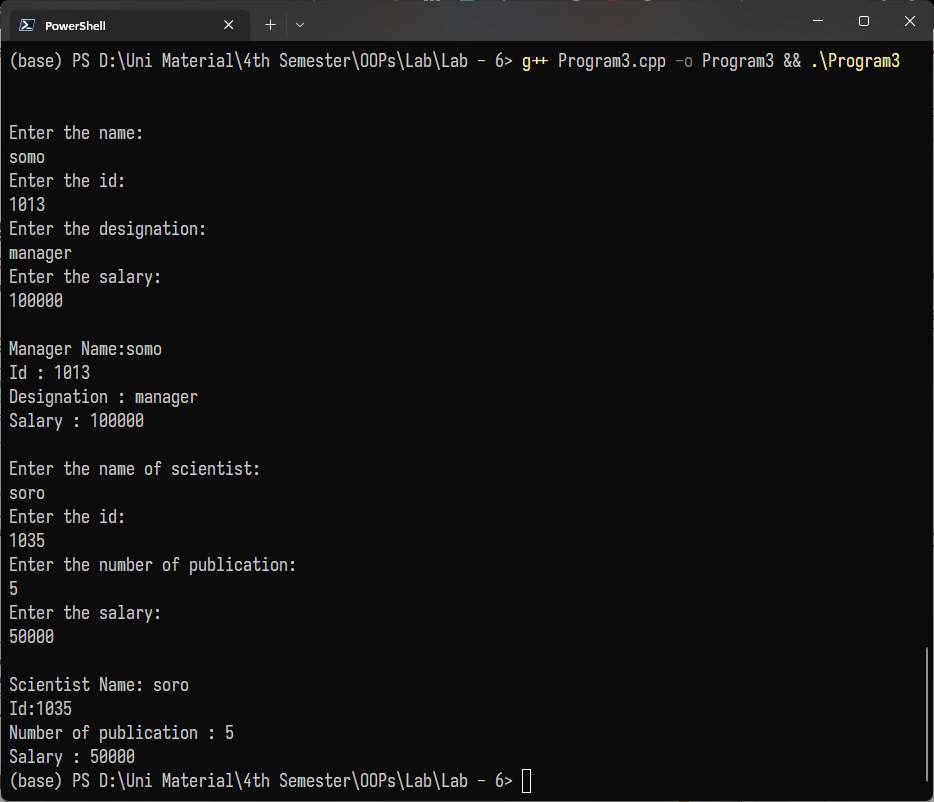
s.getdata();

s.show();

return 0;

}

Output :



**Assignment – 7**

Problem Statement :

1.Write a program to create the following class hierarchy. Create a base class called Figure and two derived classes Closed and Open from this base class. Create two more classes called Polygon and Ellipse derived from the Closed class. Create derived class Line from Open class. Define three objects (Polygon p, Ellipse e and Line l). All classes must have a constructor and a destructor which print appropriate messages. Create and then destroy the three objects. Observe how the constructors and destructors are called.

Program :

#include <iostream>

using namespace std;

class figure{

public:

figure(){

cout << "figure constructor called" << endl;

}

~figure(){

cout << "figure destructor called" << endl;

}

};

class open : public figure{

public:

open(){

cout << "open constructor called" << endl;

}

~open(){

cout << "open destructor called" << endl;

}

};

class closed : public figure{

public:

closed(){

cout << "closed constructor called" << endl;

}

~closed(){

cout << "closed destructor called" << endl;

}

};

class polygon : public closed{

public:

polygon(){

cout << "polygon constructor called" << endl;

}

~polygon(){

cout << "polygon destructor called" << endl;

}

};

class ellipse : public closed{

public:

ellipse(){

cout << "ellipse constructor called" << endl;

}

~ellipse(){

cout << "ellipse destructor called" << endl;

}

};

class line : public open{

public:

line(){

cout << "line constructor called" << endl;

}

~line(){

cout << "line destructor called" << endl;

}

};

int main(){

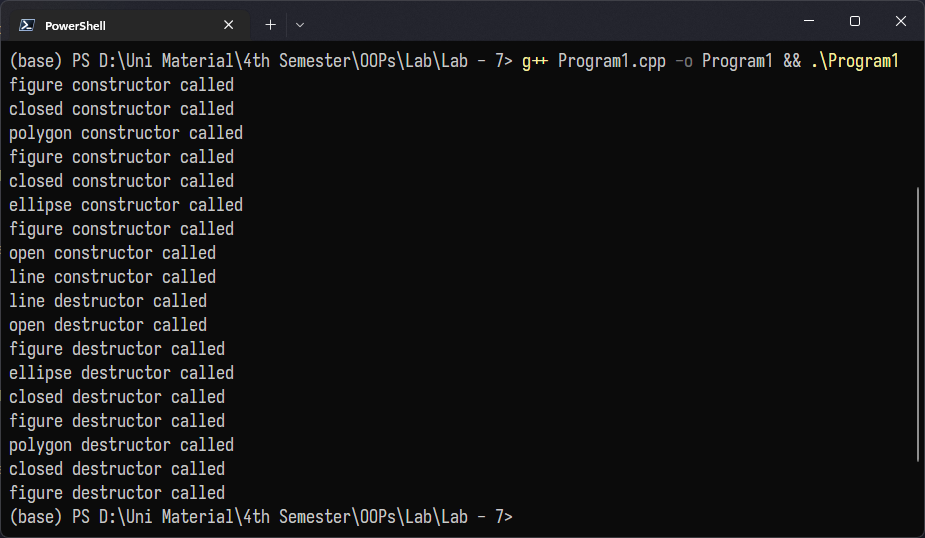
polygon p;

ellipse e;

line l;

}

Output :



Problem Statement :

2.Write a program to implement a base class consumer and a derived class transaction. Write a display method in the base class. Create an object of the derived class and call display using this object. Now, write a display method in the derived class as well. Demonstrate that calling display() using an object of the derived class, overrides the display method of the base class.

Program :

#include <iostream>

using namespace std;

class consumer

{

public:

void display()

{

cout << "consumer display";

}

};

class transaction : public consumer

{

public:

void display()

{

cout << "transaction display";

}

};

int main()

{

transaction t;

t.display();

return 0;

}

Output :

