|  |  |
| --- | --- |
| **Name:** | Divya Dipak Unkule |
| **RollNo:** | 243142 |
| **PRN No.:** | 240341220056 |
| **Subject:** | CPP Assignment |
| **Course:** | PG-DAC |

**Lab1**

1:write program to test Hello World.

Program:

/\*write program to test Hello World.\*/

#include<iostream>

using namespace std;

int main()

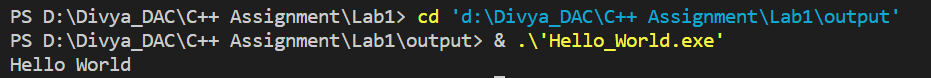
{

    cout<<"Hello World";

    return 0;

}

Output:



2:Write a program to adddition of two numbers .

Program:

/\*Write a program to adddition of two numbers .\*/

#include<iostream>

using namespace std;

int main()

{

   int n1,n2;

   cout<<"Enter Two Numbers"<<endl;

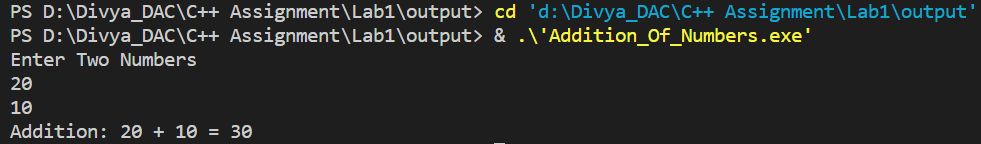
   cin>>n1>>n2;

   cout<<"Addition: "<<n1<<" + "<<n2<<" = "<<(n1+n2);

    return 0;

}

Output:



3:Write a program to swap two numbers.

Program:

/\*Write a program to swap two numbers.\*/

#include<iostream>

using namespace std;

int main()

{

    int n1,n2,temp;

   cout<<"Enter Two Numbers"<<endl;

   cin>>n1>>n2;

    cout<<"Before Swap:\n"<<"n1: "<<n1<<"\tn2: "<<n2<<endl;

    temp=n1;

    n1=n2;

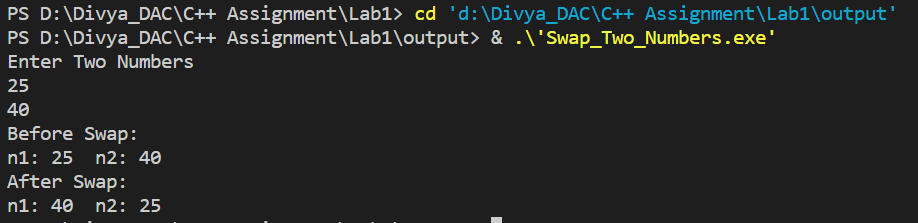
    n2=temp;

    cout<<"After Swap:\n"<<"n1: "<<n1<<"\tn2: "<<n2<<endl;

    return 0;

}

Output:



4. Write a program to accept an integer and check if it is even or odd.

Program:

/\* Write a program to accept an integer and check if it is even or odd.\*/

#include<iostream>

using namespace std;

int main()

{

   int n1;

   cout<<"Enter a number"<<endl;

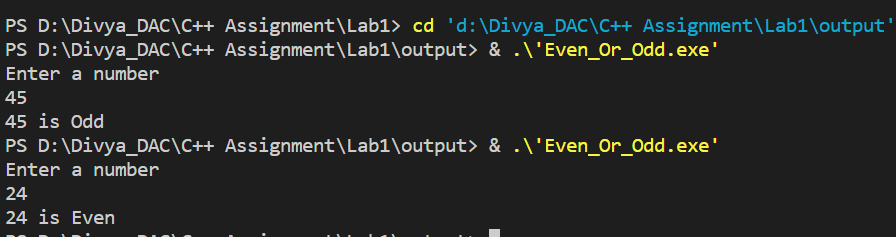
   cin>>n1;

   n1%2==0?cout<<n1<<" is Even":cout<<n1<<" is Odd";

    return 0;

}

Output:



5. Write a program to accept a number and check if it is divisible by 5 and 7.

Program:

/\*Write a program to accept a number and check if it is divisible by 5 and 7.\*/

#include<iostream>

using namespace std;

int main()

{

   int n1;

   cout<<"Enter a number"<<endl;

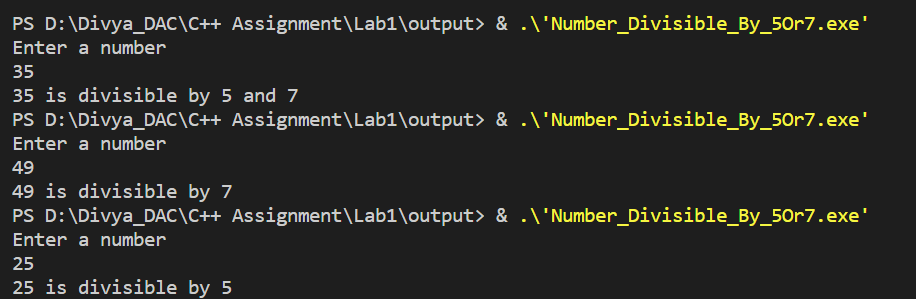
   cin>>n1;

   n1%5==0?n1%7==0?cout<<n1<<" is divisible by 5 and 7":cout<<n1<<" is divisible by 5":cout<<n1<<" is divisible by 7";

    return 0;

}

Output:



6. Write a program, which accepts annual basic salary of an employee and calculates and displays the

Income tax as per the following rules.

Basic: < 1, 50,000 Tax = 0

1, 50,000 to 3,00,000 Tax = 20%

> 3,00,000 Tax = 30%

Program:

/\*Write a program, which accepts annual basic salary of an employee and calculates and displays the

Income tax as per the following rules.

Basic: < 1, 50,000 Tax = 0

 1, 50,000 to 3,00,000 Tax = 20%

 > 3,00,000 Tax = 30% \*/

#include<iostream>

using namespace std;

 int main()

 {

    double sal;

    cout<<"Enter your Salary: "<<endl;

    cin>>sal;

    if(sal<150000)

    {

        cout<<"Tax: 0"<<endl;

    }

    else if(150000<=sal<=300000)

    {

        cout<<"Tax: "<<((20\*sal)/100)<<endl;

    }

    else if(sal>300000)

    {

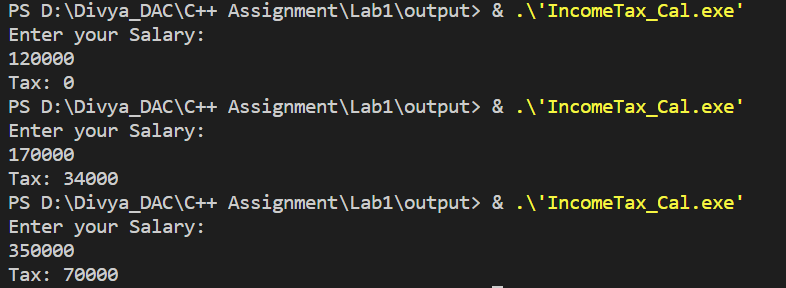
        cout<<"Tax: "<<((30\*sal)/100)<<endl;

    }

    return 0;

 }

Output:



7. Accept a lowercase character from the user and check whether the character is a vowel or consonant.

(Hint: a, e, i, o, u are vowels)

Program:

/\*Accept a lowercase character from the user and check whether the character is a vowel or consonant.\*/

#include<iostream>

using namespace std;

int main()

{

    char ch;

    cout<<"Enter lowercase character: "<<endl;

    cin>>ch;

    if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')

    {

        cout<<"charcter is vowel"<<endl;

    }

    else

    {

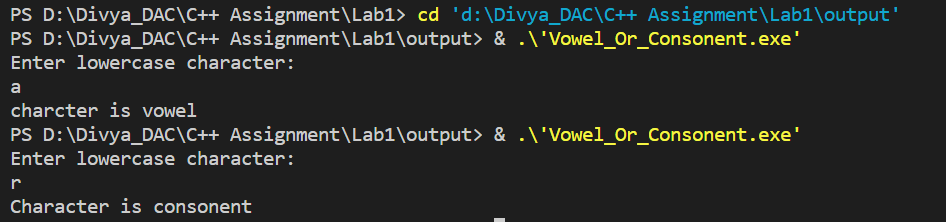
        cout<<"Character is consonent"<<endl;

    }

    return 0;

}

Output:



8. Write a program to input angles of a triangle and check whether triangle is valid or not.

Program:

//  Write a C program to input angles of a triangle and check whether triangle is valid or not.

#include<iostream>

using namespace std;

int main(void)

{

float ang1,ang2,ang3,total;

cout<<"Enter first angle of triangle"<<endl;

cin>>ang1;

cout<<"Enter second angle of triangle"<<endl;

cin>>ang2;

cout<<"Enter third angle of triangle"<<endl;

cin>>ang3;

total=ang1+ang2+ang3;

if(total==180.0f)

{

    cout<<"Triangle is valid";

}

else{

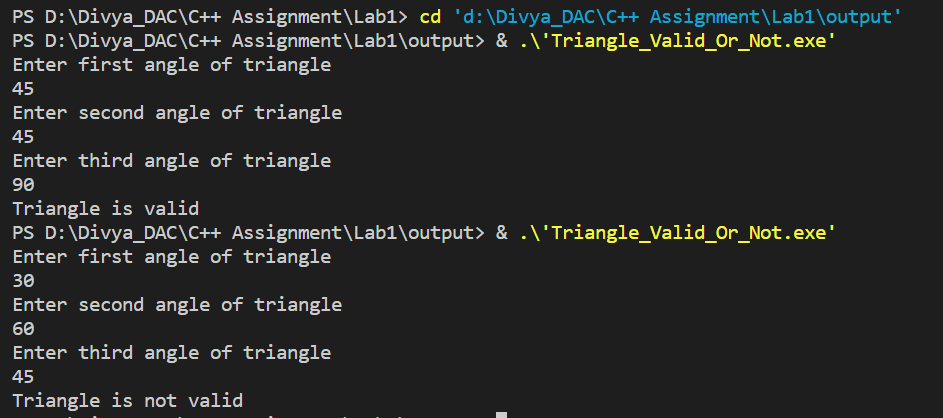
    cout<<"Triangle is not valid";

}

    return 0;

}

Output:



9:Write a program to find factorial of a given number. ex:no5 fact=5\*4\*3\*2\*1=120

Program:

// Write a program to calculate factorial of a number.

//For e.g. factorial of 5 = 5! = 5 \*4\*3\*2\*1 = 120

#include<iostream>

using namespace std;

int main(void)

{

    int num,fact=1,temp;

    cout<<"Enter any number to calculate the factorial"<<endl;

    cin>>num;

    temp=num;

    if(num!=0)

    {

    do{

        fact=fact\*num;

        num--;

    }while(num>0);

    }

    else{

        fact=1;

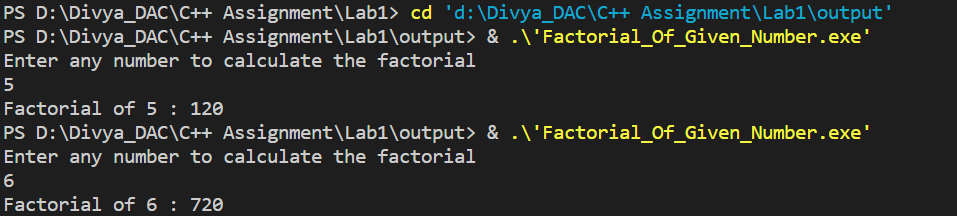
    }

    cout<<"Factorial of "<<temp<<" : "<<fact<<endl;

    return 0;

}

Output:



10:Write a program to find m to the power n. m=3 and n=4 so 3\*3\*3\*3

Program:

//Write a program to accept two integers x and n and compute x raised to n.

#include<iostream>

using namespace std;

int main(void)

{

    int x,n,pow=1;

    cout<<"Enter the number\n"<<endl;

    scanf("%d",&x);

    cout<<"Enter the power which you want to calculate for the number"<<endl;

    cin>>n;

    for(int i=1;i<=n;i++)

    {

        pow\*=x;

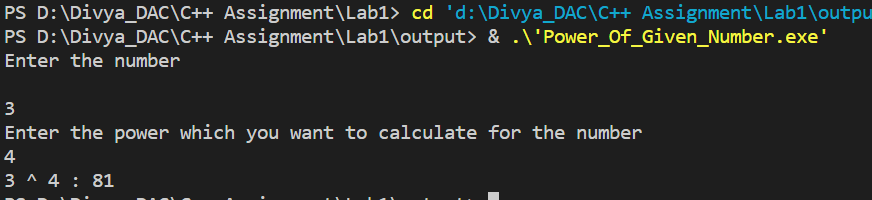
    }

    cout<<x<<" ^ "<<n<<" : "<<pow<<endl;

    return 0;

}

Output:



11:Check if number is a prime number or not.:

Program:

/\*11:Check if number is a prime number or not.: \*/

#include<iostream>

using namespace std;

int main()

{

    int num,i;

    cout<<"Enter the number"<<endl;

    cin>>num;

    for(i=2;i<num;i++)

    {

        if(num!=2 && num%i==0)

        {

            break;

        }

    }

    if(i==num)

    {

        cout<<num<<" is Prime"<<endl;

    }

    else

    {

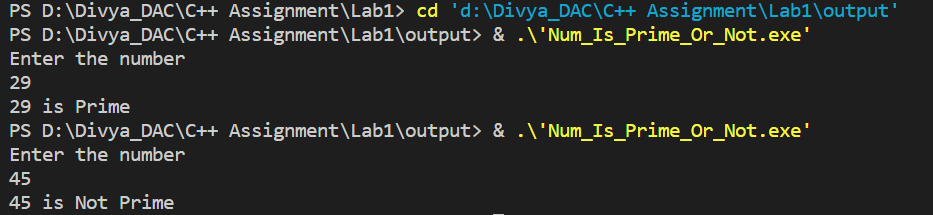
      cout<<num<<" is Not Prime"<<endl;

    }

    return 0;

}

Output:



12:Sum of series :

1+2+3+….+n

Program:

/\*12:Sum of series :

    1+2+3+….+n\*/

    #include<iostream>

    using namespace std;

    int main()

    {

        int n,sum=0;

        cout<<"Enter the number upto which you want the addition: "<<endl;

        cin>>n;

        int i=n;

        while(i>=1)

        {

            sum=sum+i;

            i--;

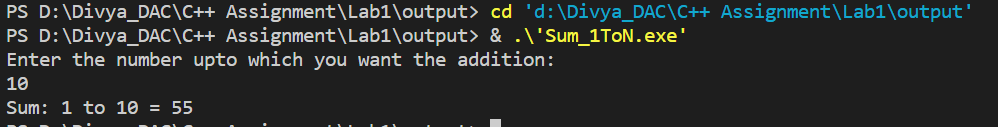
        }

        cout<<"Sum: 1 to "<< n<<" = "<<sum;

        return 0;

    }

Output:



13:Check whether the number is palindrome or not?

Program:

//13:Check whether the number is palindrome or not?

#include<iostream>

using namespace std;

int main()

{

    int num,r,n=0;

    cout<<"Enter Number"<<endl;

    cin>>num;

    int temp=num;

    while(num!=0)

    {

        r=num%10;

        n=(n\*10)+r;

        num=num/10;

    }

    if(n==temp)

    {

        cout<<temp<<" is palindrome number"<<endl;

    }

    else

    {

        cout<<temp<<" is not a palindrome number"<<endl;

    }

    return 0;

}

Output:

//13:Check whether the number is palindrome or not?

#include<iostream>

using namespace std;

int main()

{

    int num,r,n=0;

    cout<<"Enter Number"<<endl;

    cin>>num;

    int temp=num;

    while(num!=0)

    {

        r=num%10;

        n=(n\*10)+r;

        num=num/10;

    }

    if(n==temp)

    {

        cout<<temp<<" is palindrome number"<<endl;

    }

    else

    {

        cout<<temp<<" is not a palindrome number"<<endl;

    }

    return 0;

}

14:Write a program to find sum of all even and odd numbers between 1 to n.

Program:

//14:Write a  program to find sum of all even and odd numbers between 1 to n.

#include<iostream>

using namespace std;

int main()

{

    int num,sum\_eve=0,sum\_odd=0,i=1;

    cout<<"Enter NUmber upto which you want the addition of even and odd"<<endl;

    cin>>num;

     while(i<=num)

     {

        if(i%2==0)

        {

            sum\_eve=sum\_eve+i;

        }

        else if(i%2!=0)

        {

            sum\_odd=sum\_odd+i;

        }

        i++;

     }

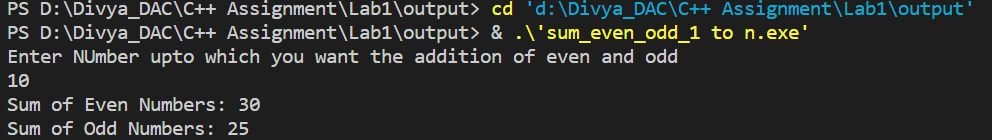
    cout<<"Sum of Even Numbers: "<<sum\_eve<<endl;

    cout<<"Sum of Odd Numbers: "<<sum\_odd<<endl;

   return 0;

}

Output:



15: Write a program to enter a number and print its reverse.

Program:

//15: Write a  program to enter a number and print its reverse.

#include<iostream>

using namespace std;

int main()

{

    int num,rev=0,r;

    cout<<"Enter a number: "<<endl;

    cin>>num;

    int temp=num;

    while(num!=0)

    {

       r=num%10;

       rev=(rev\*10)+r;

       num=num/10;

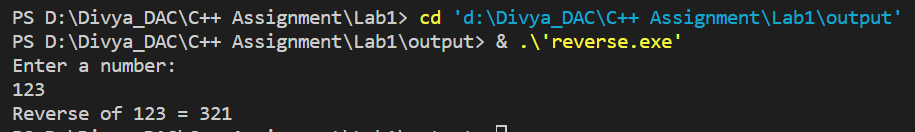
    }

    cout<<"Reverse of "<<temp<<" = "<<rev<<endl;

    return 0;

}

Output:



16:Write a program to print all Prime numbers between 1 to n.

Program:

//16:Write a  program to print all Prime numbers between 1 to n.

#include<iostream>

using namespace std;

int main()

{

    int num,i=1,j;

    cout<<"Enter a number"<<endl;

    cin>>num;

    cout<<"Prime Numbers from 1 to "<<num<<endl;

    while(i<=num)

    {

        for(j=2;j<num;j++)

        {

            if(i!=2 && i%j==0)

            {

                break;

            }

        }

        if(i==2 || j==i)

        {

            cout<<i<<" ";

        }

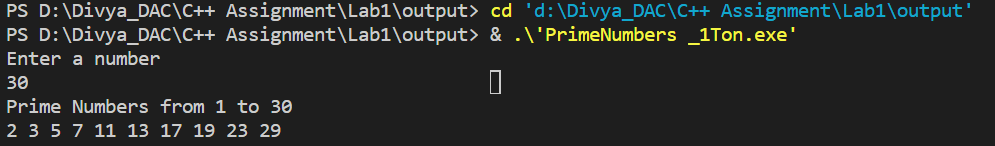
        i++;

    }

    return 0;

}

Output:



17:Write a program to check entered number is Armstrong number or not.

Program:

//17:Write a program to check entered number is Armstrong number or not.

#include<iostream>

using namespace std;

int main()

{

    int num,temp,temp1,sum=0,d=0,mul,r;

    cout<<"Enter Number: "<<endl;

    cin>>num;

    temp=num;

    temp1=num;

    while(num!=0)

    {

        d++;

        num=num/10;

    }

    while(temp!=0)

    {

        r=temp%10;

        mul=1;

        for(int i=1;i<=d;i++)

        {

            mul=mul\*r;

        }

        sum=sum+mul;

        temp=temp/10;

    }

    if(sum==temp1)

    {

        cout<<temp1<<" is armstrong number"<<endl;

    }

    else

    {

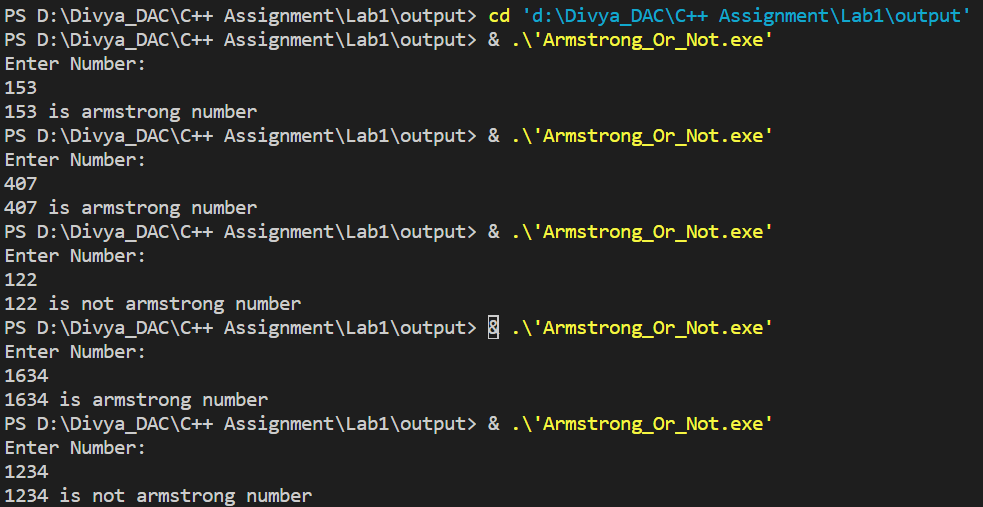
        cout<<temp1<<" is not armstrong number"<<endl;

    }

    return 0;

}

Output:



18:Write a program to find greatest of three numbers using nested if-else.

Program:

//18:Write a program to find greatest of three numbers using nested if-else.

#include<iostream>

using namespace std;

int main()

{

    int n1,n2,n3;

    cout<<"Enter three Numbers"<<endl;

    cin>>n1>>n2>>n3;

    if(n1>n2)

    {

        if(n1>n3)

        {

            cout<<n1<<" is Greatest number"<<endl;

        }

        else

        {

            cout<<n3<<" is Greatest number"<<endl;

        }

    }

    else if(n2>n3)

    {

       cout<<n2<<" is Greatest number"<<endl;

    }

    else

    {

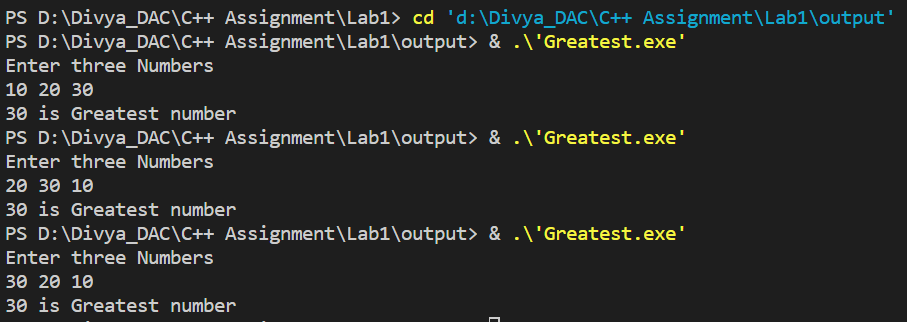
        cout<<n3<<" is Greatest number"<<endl;

    }

    return 0;

}

Output:



19:Create menu driven program for Pizza Shop.And display total amount

Program:

//19:Create menu driven program for Pizza Shop.And display total amount

#include <iostream>

using namespace std;

int displayMenu() {

    cout << "Welcome to Our Pizza Shop!" << endl;

    cout << "1. Margherita Pizza - 100/- " << endl;

    cout << "2. Pepperoni Pizza - 150/-" << endl;

    cout << "3. Vegetarian Pizza - 90/-" << endl;

    cout << "4. Hawaiian Pizza - 250/-" << endl;

    cout << "5. Exit" << endl;

    cout << "Enter your choice (1-5): ";

    int choice;

    cin >> choice;

    return choice;

}

double calculateTotal(int choice) {

    double price;

    switch(choice) {

        case 1:

            price = 100;

            break;

        case 2:

            price = 150;

            break;

        case 3:

            price = 90;

            break;

        case 4:

            price = 250;

            break;

        default:

            price = 0;

    }

    return price;

}

int main() {

    double total = 0;

    int choice;

    do {

        choice = displayMenu();

        if(choice >= 1 && choice <= 4) {

            total += calculateTotal(choice);

            cout << "Added to your cart!" << endl;

        } else if(choice != 5) {

            cout << "Invalid choice. Please enter a number between 1 and 5." << endl;

        }

    } while(choice != 5);

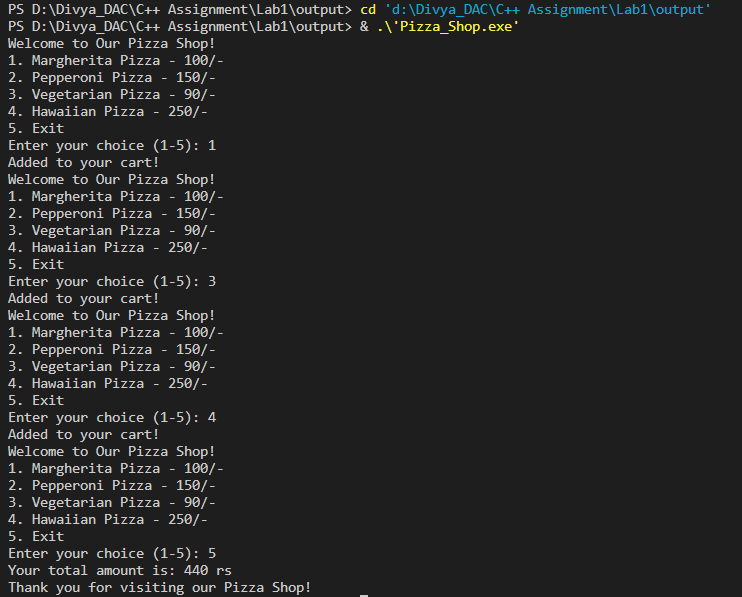
    cout << "Your total amount is: " << total <<" rs"<< endl;

    cout << "Thank you for visiting our Pizza Shop!" << endl;

    return 0;

}

Output:



20:Accept a single digit from the user and display it in words. For example, if digit entered is 9, display Nine.

Program:

// Accept a single digit from the user and display it in words. For example, if digit entered is 9, display Nine.

#include<iostream>

using namespace std;

int main(void)

{

    int digit;

    cout<<"Enter any sigle digit"<<endl;

    cin>>digit;

    switch(digit)

    {

        case 1:

            cout<<"digit entered: "<<digit<<" - One"<<endl;

            break;

        case 2:

            cout<<"digit entered: "<<digit<<" - Two"<<endl;

            break;

        case 3:

           cout<<"digit entered: "<<digit<<" - Three"<<endl;

            break;

        case 4:

            cout<<"digit entered: "<<digit<<" - Four"<<endl;

             break;

        case 5:

            cout<<"digit entered: "<<digit<<" - Five"<<endl;

             break;

        case 6:

            cout<<"digit entered: "<<digit<<" - Six"<<endl;

             break;

        case 7:

            cout<<"digit entered: "<<digit<<" - Seven"<<endl;

             break;

        case 8:

           cout<<"digit entered: "<<digit<<" - Eight"<<endl;

            break;

        case 9:

            cout<<"digit entered: "<<digit<<" - Nine"<<endl;

            break;

        default:

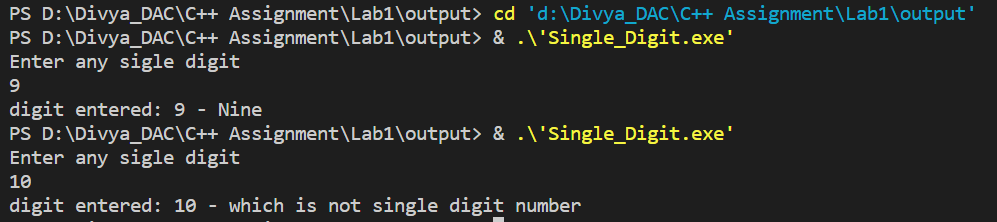
            cout<<"digit entered: "<<digit<<" - which is not single digit number"<<endl;

    }

    return 0;

}

Output:



21. Write a program, which accepts two integers and an operator as a character (+ - \* / ), performs the corresponding operation and displays the result.

Program:

/\*Write a program, which accepts two integers and an operator as a character (+ - \* / ), performs the

corresponding operation and displays the result.\*/

#include<iostream>

using namespace std;

int main(void)

{

    int num1,num2;

    char opr;

    cout<<"Enter any two numbers"<<endl;

    cin>>num1>>num2;

    cout<<"Any operation which you want to perform, enter the operator for that"<<endl;

    fflush(stdin);

    cin>>opr;

    switch(opr)

    {

        case '+':

            cout<<num1 <<opr<< num2<< "= "<<(num1+num2);

            break;

        case '-':

           cout<<num1 <<opr<< num2<< "= "<<(num1-num2);

            break;

        case '\*':

            cout<<num1 <<opr<< num2<< "= "<<(num1\*num2);

            break;

        case '/':

           cout<<num1 <<opr<< num2<< "= "<<(num1/num2);

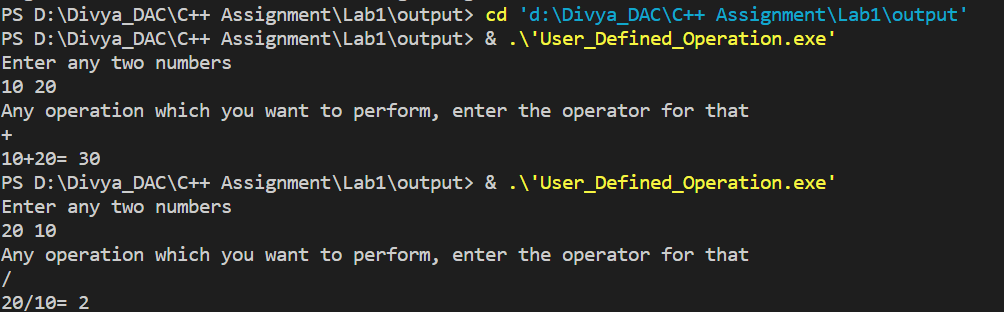
            break;

    }

    return 0;

}

Output:



**Lab2**



1:Write a program that accepts numbers continuously as long as the number is positive and prints the sum of the given numbers.

Program:

/\*Write a program that accepts numbers continuously as long as the number is positive and prints the

sum of the given numbers. \*/

#include<iostream>

using namespace std;

int main()

{

    int sum=0,num=0;

    do{

        cout<<"Enter Positive Integer"<<endl;

        cin>>num;

        if(num>0)

        {

        sum=sum+num;

        }

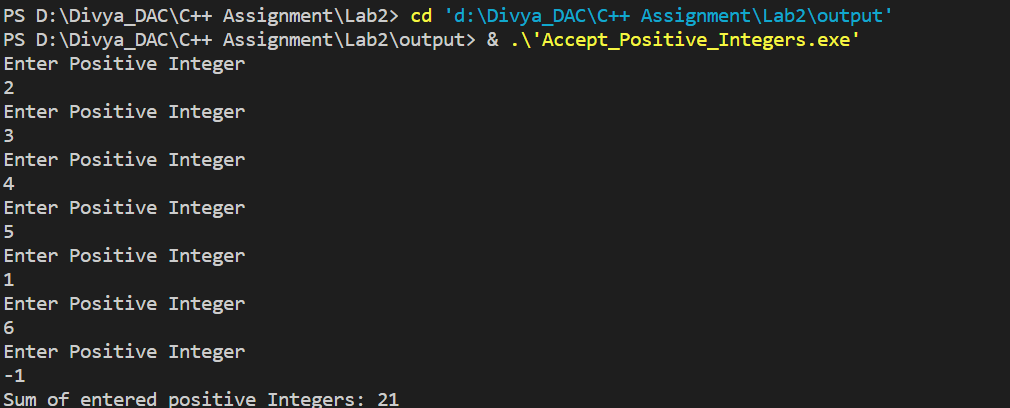
    }while(num>0);

    cout<<"Sum of entered positive Integers: "<<sum<<endl;

    return 0;

}

Output:



2. Write a program to accept two integers x and n and compute x raised to n.

Program:

/\*Write a program to accept two integers x and n and compute x raised to n.\*/

#include<iostream>

using namespace std;

int main()

{

    int x,n,pow=1,temp;

    cout<<"Enter number"<<endl;

    cin>>x;

    cout<<"Enter power"<<endl;

    cin>>n;

    temp=n;

     while(n!=0)

     {

        pow=pow\*x;

        n--;

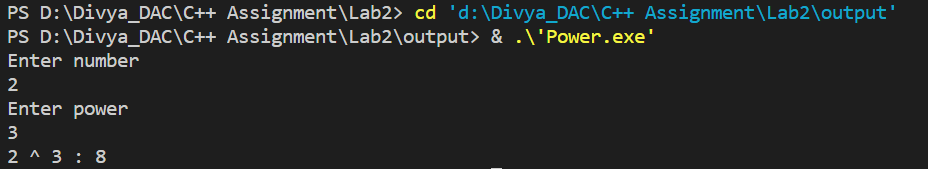
     }

    cout<<x<<" ^ "<<temp<<" : "<<pow<<endl;

    return 0;

}

Output:



3. Write a program to accept a character, an integer n and display the next n characters.

Program:

/\*Write a program to accept a character, an integer n and display the next n characters.\*/

#include<iostream>

using namespace std;

int main()

{

    char ch;

    int n;

    cout<<"Enter a character"<<endl;

    cin>>ch;

    cout<<"Enter nunber upto which you want to print next characters"<<endl;

    cin>>n;

    while(n!=0)

    {

        cout<<char(ch+1)<<endl;

        ch++;

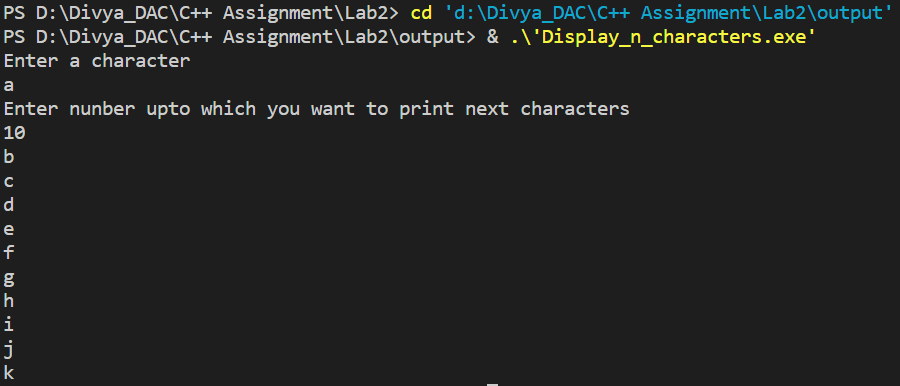
        n--;

    }

    return 0;

}

Output:



4. Write a program to calculate factorial of a number.

For e.g. factorial of 5 = 5! = 5 \*4\*3\*2\*1 = 120

Program:

/\*Write a program to calculate factorial of a number.

For e.g. factorial of 5 = 5! = 5 \*4\*3\*2\*1 = 120\*/

#include<iostream>

using namespace std;

int fact(int n)

{

    int f=1,temp=n;

    if(n==0)

    {

        return 1;

    }

    f=temp\*fact(--n);

    return f;

}

int main()

{

    int n,factorial;

    cout<<"Enter an integer"<<endl;

    cin>>n;

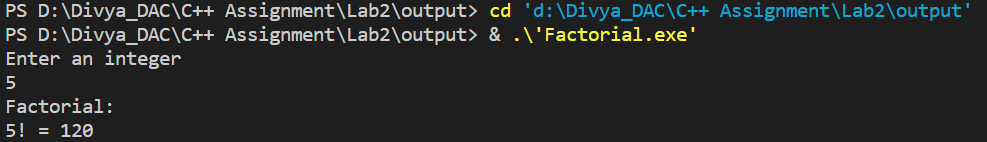
    factorial=fact(n);

    cout<<"Factorial:\n"<<n<<"! = "<<factorial<<endl;

    return 0;

}

Output:



5. Write a program to calculate factors of a given number.

Program:

/\*Write a program to calculate factors of a given number\*/

#include<iostream>

using namespace std;

int main(void)

{

    int num;

    cout<<"Enter any number"<<endl;

    cin>>num;

    cout<<"Factors of "<<num<<endl;

    for(int i=1;i<=num;i++)

    {

        if(num%i==0)

        {

            cout<<i<<endl;

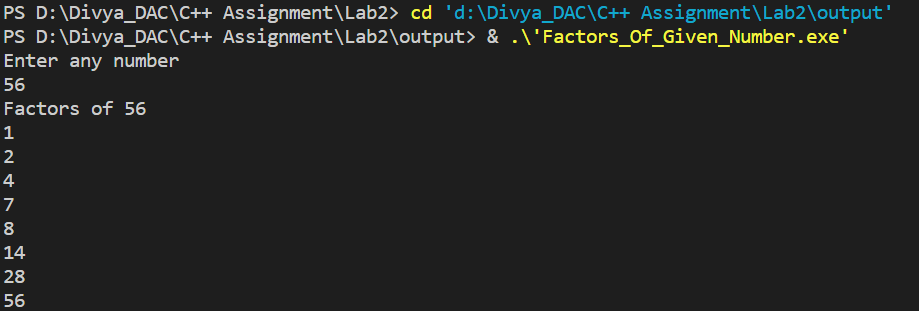
        }

    }

    return 0;

}

Output:



6. Accept two numbers and calculate GCD of them.

Program:

/\*Accept two numbers and calculate GCD of them.\*/

#include<iostream>

using namespace std;

int main(void)

{

    int n1,n2;

    cout<<"Enter two numbers to calculate GCD:"<<endl;

    cin>>n1>>n2;

    int GCD=1;

    if(n1>n2)

    {

       for(int i=1;i<n2;i++)

       {

          if(n1%i==0 && n2%i==0)

          {

            if(i>=GCD)

            {

                GCD=i;

            }

          }

       }

    }

    else{

         for(int i=1;i<n1;i++)

       {

          if(n1%i==0 && n2%i==0)

          {

            if(i>=GCD)

            {

                GCD=i;

            }

          }

       }

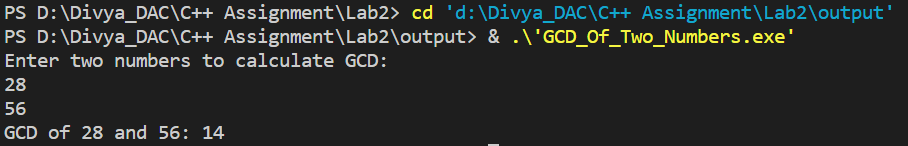
    }

    cout<<"GCD of "<<n1<<" and "<<n2<<": "<<GCD;

    return 0;

}

Output:



7. Write a menu driven program to do following operations :

a) Compute area of circle

b) Compute area of rectangle

c) Compute area of triangle

d) Exit

Display menu, ask choice to the user, depending on choice accept the parameters and perform the

operation. Continue this process until user selects exit option.

Program:

/\*Write a menu driven program to do following operations :

a) Compute area of circle

b) Compute area of rectangle

c) Compute area of triangle

d) Exit

Display menu, ask choice to the user, depending on choice accept the parameters and perform the

operation. Continue this process until user selects exit option.\*/

#include<iostream>

using namespace std;

int main(void)

{

    int choice;

    for(;1;)

    {

        cout<<"Enter the choice from following:\n1) Compute area of circle\n2) Compute area of rectangle\n3) Compute area of triangle\n0) Exit \n";

        cin>>choice;

        switch(choice)

        {

            case 1:

            {

                float r;

                cout<<"Enter radius of circle\n";

                cin>>r;

                cout<<"Area of circle:"<<(3.14\*r\*r)<<endl;

                continue;

            }

             case 2:

            {

                float l,w;

                cout<<"Enter length of rectangle\n";

                cin>>l;

                cout<<"Enter width of rectangle\n";

                cin>>w;

                cout<<"Area of rectangle:"<<(l\*w)<<endl;

                continue;

            }

             case 3:

            {

               float b,h;

                cout<<"Enter breadth of triangle\n";

                cin>>b;

                cout<<"Enter height of triangle\n";

                cin>>h;

                cout<<"Area of triangle: "<<((b\*h)/2)<<endl;

                continue;

            }

             case 0:

            {

                return 0;

            }

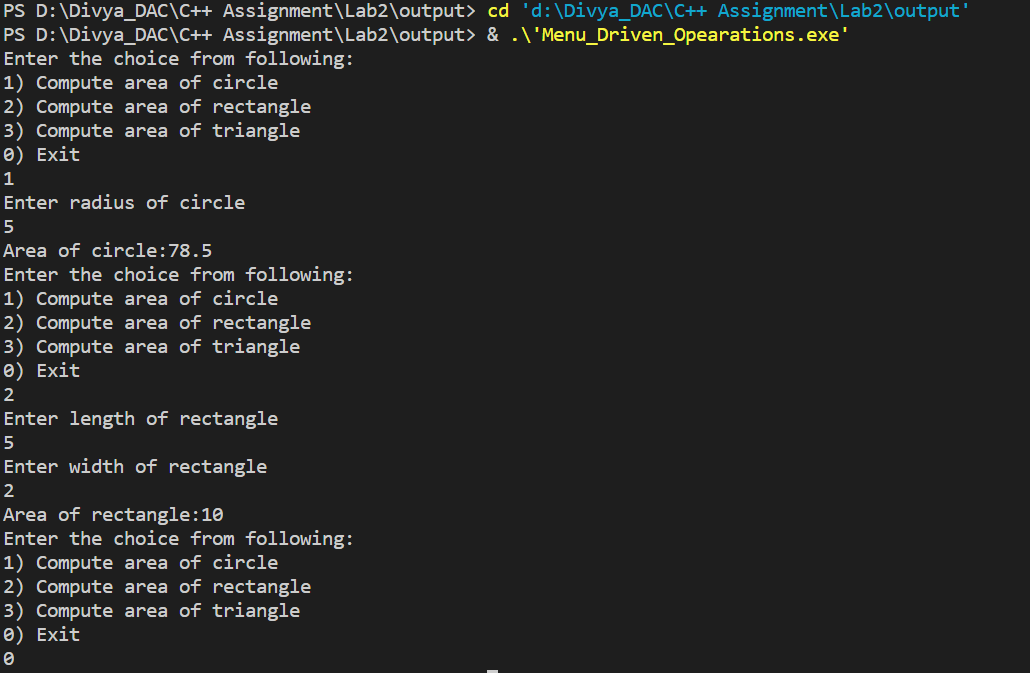
        }

    }

    return 0;

}

Output:



8. Write a program to print all prime numbers between 1 to n

Program:

/\*Write a program to print all prime numbers between 1 to n\*/

#include<iostream>

using namespace std;

void isPrime(int i)

{

    int flag=0;

    for(int j=2;j<i;j++)

    {

        if(i!=2 && i%j==0)

        {

            flag=1;

            break;

        }

    }

    if(flag==0)

    {

        cout<<i<<endl;

    }

}

int main()

{

    int n;

    cout<<"Enter the number till which you want to display the prime numbers"<<endl;

    cin>>n;

    for(int i=2;i<=n;i++)

    {

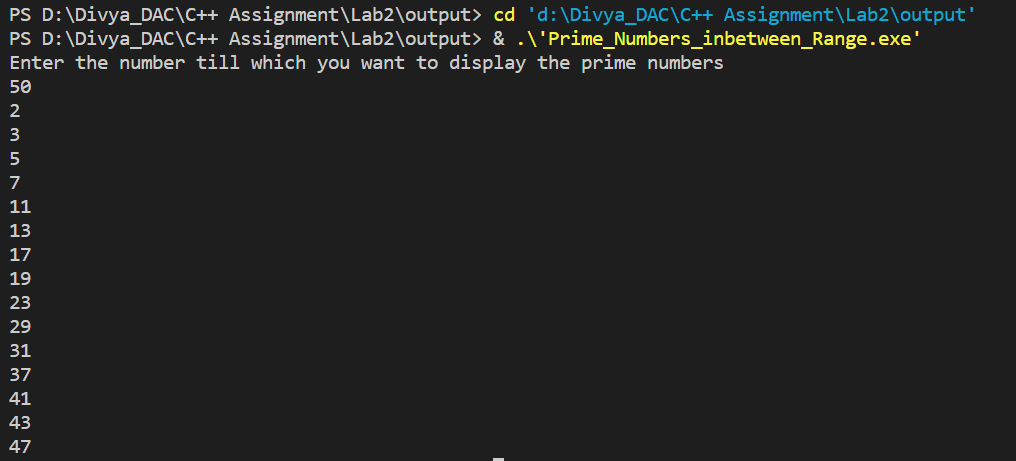
        isPrime(i);

    }

    return 0;

}

Output:



**Lab3**

1:Write a program to create an array of integers and perform following operations on that array like

finding the sum, average, maximum and minimum number in that array. Accept the numbers of the

array from user.

Program:

/\*Write a program to create an array of integers and perform following operations on that array like

finding the sum, average, maximum and minimum number in that array. Accept the numbers of the

array from user. \*/

#include<iostream>

using namespace std ;

 void sum(int arr[], int size){

  int sum = 0 ;

  for(int i = 0; i<size ; i++){

       sum = sum+arr[i];

  }

  cout<<"sum :"<<sum <<endl;

 }

 void average(int arr[], int size){

   int sum = 0 , avg ;

  for(int i = 0; i<size ; i++){

       sum = sum+arr[i];

  }

   avg = sum/size;

   cout<<"average :"<< avg <<endl;

 }

 void MaxAndMin(int arr[], int size){

  int min = arr[0];

  int max= arr[0];

  for(int i = 0 ;i<size ; i++){

    if(arr[i]>max)

      max = arr[i];

     else if(arr[i]<min)

      min = arr[i];

     }

  cout<<"Max = " <<max <<"Min = "<<min<<endl;

   }

int main(){

 int choice ;

  int size ;

  cout<<"entre size";

  cin>>size;

  int arr[size];

  cout<<"enter "<<size<<" elements"<<endl;

  for(int i = 0; i<size ; i++){

    cin>>arr[i];

  }

do

{

cout<<"Enter Your Choice: \n1.sum\n2.average\n3.MaxandMin\n5.exit"<<endl;

cin>>choice;

switch(choice)

{

    case 1:

    sum(arr,size);

   break;

    case 2:

    average(arr,size);

 break;

    case 3:

    MaxAndMin(arr,size);

break;

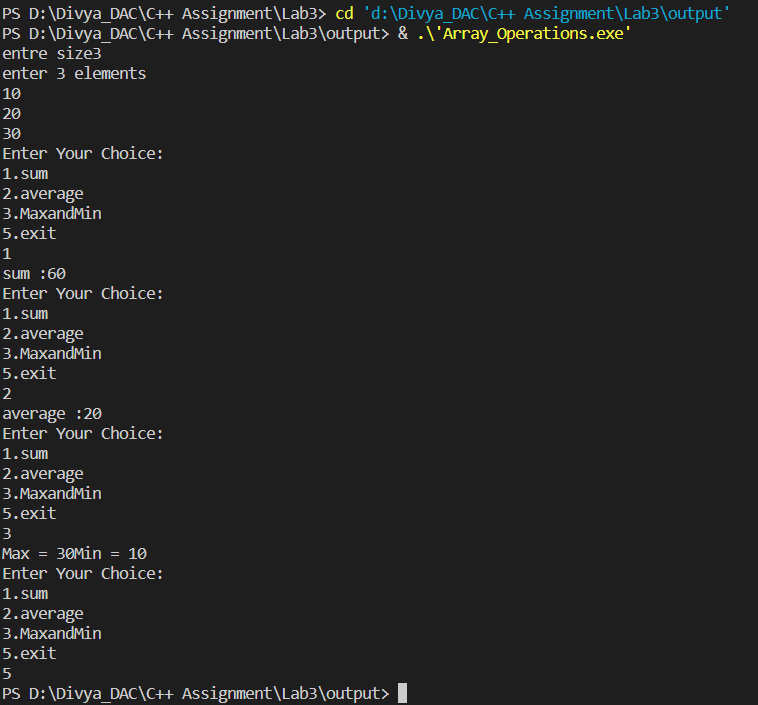
}

}while(choice!=5);

    return 0;

}

Output:



2: Write a program to Accept a number and display its sum of digits.:ex 568 5+6+8=19

Program:

/\*Write a program to Accept a number and display its sum of digits.:ex 568    5+6+8=19\*/

#include<iostream>

using namespace std;

int main()

{

    int num,sum=0;

    cout<<"Enter Number: "<<endl;

    cin>>num;

    while(num>0)

    {

        sum=sum+(num%10);

        num=num/10;

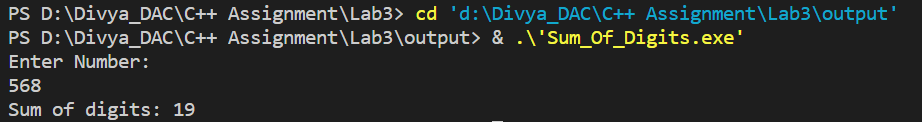
    }

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

return 0;

}

Output:



3:. Write a program to find sum of all even and odd numbers between 1 to n.

Program:

// Write a  program to find sum of all even and odd numbers between 1 to n

#include<iostream>

using namespace std ;

int main(){

    int n ,sum\_Even=0 , sum\_Odd =0;

     cout<<"Enter the last digit"<<endl;

     cin>>n;

    while(n>0){

     if(n%2==0){

        sum\_Even+=n;

     }

     else {

        sum\_Odd = sum\_Odd + n ;

     }

     n-- ;

    }

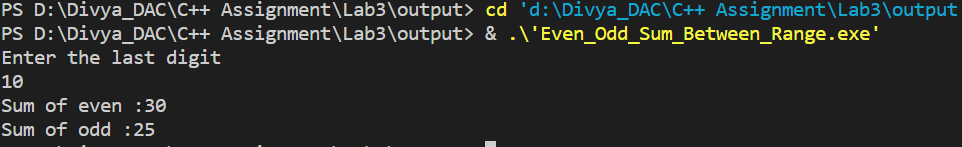
    cout<<"Sum of even :" <<sum\_Even <<endl;

    cout<<"Sum of odd :" <<sum\_Odd<<endl ;

    return 0 ;

}

Output:



4:. Write a program to print all Prime numbers between 1 to n.

Program:

  // Write a  program to print all Prime numbers between 1 to n.

  #include<iostream>

  using namespace std ;

  void isprime(int n){

    int flag = 0 ;

    for(int i=2; i<n;i++){

        if(n!=2 && n%i==0)

        {

             flag = 1 ;

             break;

        }

     }

    if(n!=1 && flag==0){

      cout<<n<<" ";

      }

  }

  int main(){

    int n ;

    cout<<"enter the last digit";

    cin>>n;

    for(int i = 1 ; i<=n;i++){

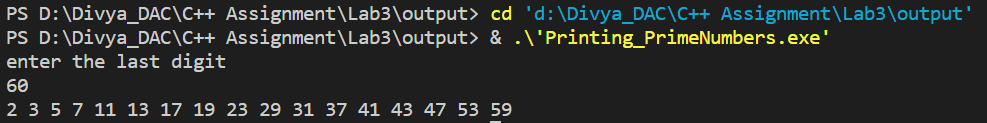
      isprime(i);

      }

    return 0 ;

  }

Output:



5:Write a program to accept array from user .Accept number from user and search number is present in array or not.

Program:

/\*Write a program to accept array  from user .Accept number from user and search number is present in array or not.\*/

#include<iostream>

using namespace std;

void binary\_Search(int arr[],int key,int size)

{

    int right=size-1,left=0,mid,i;

    for(i=0;i<size;i++)

    {

        mid=(left+right)/2;

        if(key==arr[mid])

        {

            cout<<key<<"found at "<<mid+1<<" position"<<endl;

            break;

        }

       else if(key>arr[mid])

        {

        left=mid+1;

        }

       else if(key<arr[mid])

       {

        right=mid-1;

       }

    }

    if(i==size)

    {

        cout<<key<<"is not present in an array"<<endl;

    }

}

int main()

{

int size,key;

  cout<<"entre size";

  cin>>size;

  int arr[size];

  cout<<"enter "<<size<<" elements"<<endl;

  for(int i = 0; i<size ; i++){

    cin>>arr[i];

  }

  cout<<"Eneter element to search: "<<endl;

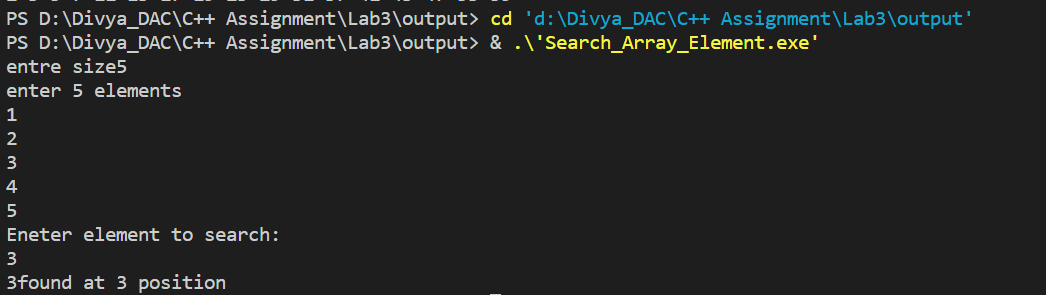
  cin>>key;

  binary\_Search(arr,key,size);

  return 0;

}

Output:



6:Write a program to print following pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

Program:

/\*Write a program to print following pattern.\*/

#include<iostream>

using namespace std;

int main()

{

    for(int i=1;i<=5;i++)

    {

        int temp=i;

        while(temp>0)

        {

            cout<<"\* ";

            temp--;

        }

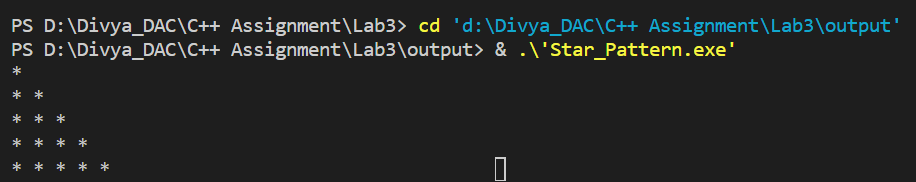
        cout<<endl;

    }

    return 0;

}

Output:



7:Write a program to create student class with data members rollno, marks1,mark2,mark3.

Accept data (acceptInfo()) and display using display member function.

Also display total,percentage and grade.

Program:

/\*Write a program to create student class with data members rollno, marks1,mark2,mark3.

Accept data (acceptInfo()) and display  using display member function.

Also display total,percentage and grade.\*/

#include<iostream>

using namespace std ;

class student{

    private :

     int Roll\_No ;

     int    M\_Maths ;

    int M\_Science;

    int M\_English;

 public :

 void acceptInfo(){

   cout<<"Enter the rollno , maths marks , science marks , english marks "<<endl;

   cin>>Roll\_No>>M\_Maths>>M\_Science>>M\_English;

 }

 void display(){

    cout<<"Roll no :"<<Roll\_No;

    cout<<" Maths Marks :"<<M\_Maths;

    cout<<"Science Marks :"<<M\_Science;

    cout<<"English Marks:"<<M\_English;

 }

 int getTotal(){

    int total ;

    total = M\_Maths+M\_Science+M\_English;

    return total ;

 }

  float getPercentage(){

    float percent = (getTotal() \* 100) / 300;

    return percent;

 }

};

int main(){

    int total ;

    float percent ;

 student s ;

        s.acceptInfo();

    s.display();

 total= s.getTotal();

 percent=s.getPercentage();

 cout<<"total : "<<total<<endl;

 cout<<"percent : "<<percent<<endl;

 if(percent  >=90.00)

    cout<<"Grade A";

    else if(percent<=90.00 && percent>=80.00)

    cout<<"Grade B";

    else if(percent<=80.00 && percent>=60.00)

    cout<<"Grade C";

     else if(percent<=60.00 && percent>=40.00)

    cout<<"Grade D";

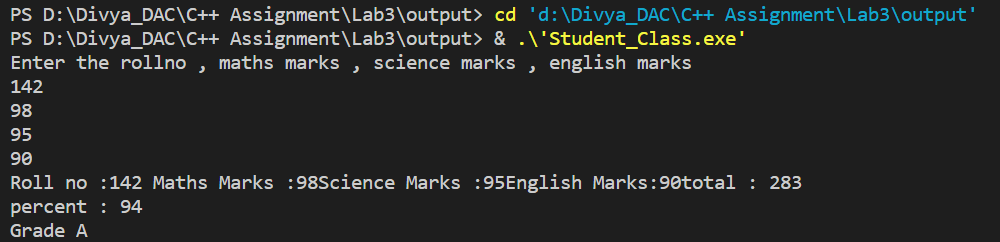
    else

    cout<<"Fail";

 return 0 ;

}

Output:



**Lab4**



1.Write a program to create student class with data members rollno, marks1,mark2,mark3.

Accept data (acceptInfo()) and display using display member function.

Also display total,percentage and grade.

Program:

/\*1:Write a program to create student class with data members rollno, marks1,mark2,mark3.

Accept data (acceptInfo()) and display  using display member function.

Also display total,percentage and grade.

\*/

#include<iostream>

using namespace std;

class Student

{

    private:

    int rollNo;

    int marks1;

    int marks2;

    int marks3;

     int total()

     {

        return (marks1+marks2+marks3);

     }

     float percentage()

     {

         return (total()\*100)/300;

     }

    public:

    void acceptInfo(int rollNo,int marks1,int marks2,int marks3)

    {

       this->rollNo=rollNo;

       this->marks1=marks1;

       this->marks2=marks2;

       this->marks3=marks3;

    }

    void displayInfo()

    {

       cout<<"rollNo: "<<rollNo<<endl;

       cout<<"marks1: "<<marks1<<endl;

       cout<<"marks2: "<<marks2<<endl;

       cout<<"marks3: "<<marks3<<endl;

       cout<<"Your Total marks out of 300: "<<total()<<endl;

       cout<<"Your Percentage: "<<percentage()<<endl;

    }

    char getGrade()

    {

       if(percentage()>=90)

       {

          return 'A';

       }

       else if(percentage()<=90 && percentage()>=70)

       {

          return 'B';

       }

       else if(percentage()<=70 && percentage()>=60)

       {

          return 'C';

       }

       else if(percentage()<=60 && percentage()>=40)

       {

          return 'D';

       }

    }

};

int main()

{

   int rollNo;

    int marks1;

    int marks2;

    int marks3;

   Student s1;

   cout<<"Enter \n1.rollNo\n2.marks1\n2.marks2\n3.marks3"<<endl;

   cin>>rollNo>>marks1>>marks2>>marks3;

s1.acceptInfo(rollNo,marks1,marks2,marks3);

cout<<"Your Information:"<<endl;

s1.displayInfo();

cout<<"Your Grade: "<<s1.getGrade()<<endl;

    return 0;

}

Output:



2. Create a class Person with data members as name, age, city. Write getters and setters for all the data members. Also add the display function. Create Default and Parameterized constructors. Create the object of this class in main method and invoke all the methods in that class.

Program:

#include<bits/stdc++.h>

using namespace std;

class Person {

private:

    char\* name;

    int age;

    char\* city;

public:

    Person() {

        name = new char[1];

        \*name = '\0';

        age = 0;

        city = new char[1];

        \*city = '\0';

    }

    Person(const char\* name, int age, const char\* city) {

        this->name = new char[strlen(name) + 1];

        strcpy(this->name, name);

        this->age = age;

        this->city = new char[strlen(city) + 1];

        strcpy(this->city, city);

    }

    ~Person() {

        delete[] name;

        delete[] city;

    }

    const char\* getName() const {

        return name;

    }

    int getAge() const {

        return age;

    }

    const char\* getCity() const {

        return city;

    }

    void setName(const char\* name) {

        delete[] this->name;

        this->name = new char[strlen(name) + 1];

        strcpy(this->name, name);

    }

    void setAge(int age) {

        this->age = age;

    }

    void setCity(const char\* city) {

        delete[] this->city;

        this->city = new char[strlen(city) + 1];

        strcpy(this->city, city);

    }

    void displayInfo() const {

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

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

        cout << "City: " << city << endl;

    }

};

int main() {

    Person p1("John", 30, "New York");

    p1.displayInfo();

    int choice;

    do {

        cout << "Enter your choice\n1. Set Name\n2. Get Name\n3. Set Age\n4. Get Age\n5. Set City\n6. Get City\n7. Display Info\n8. Exit" << endl;

        cin >> choice;

        switch (choice) {

            case 1: {

                char name[100];

                cout << "Enter your name: ";

                cin >> name;

                p1.setName(name);

                break;

            }

            case 2: {

                cout << "Name: " << p1.getName() << endl;

                break;

            }

            case 3: {

                int age;

                cout << "Enter your age: ";

                cin >> age;

                p1.setAge(age);

                break;

            }

            case 4: {

                cout << "Age: " << p1.getAge() << endl;

                break;

            }

            case 5: {

                char city[100];

                cout << "Enter your city: ";

                cin >> city;

                p1.setCity(city);

                break;

            }

            case 6: {

                cout << "City: " << p1.getCity() << endl;

                break;

            }

            case 7: {

                p1.displayInfo();

                break;

            }

            case 8: {

                cout << "Exiting..." << endl;

                break;

            }

            default: {

                cout << "Invalid choice. Please enter a number between 1 and 8." << endl;

                break;

            }

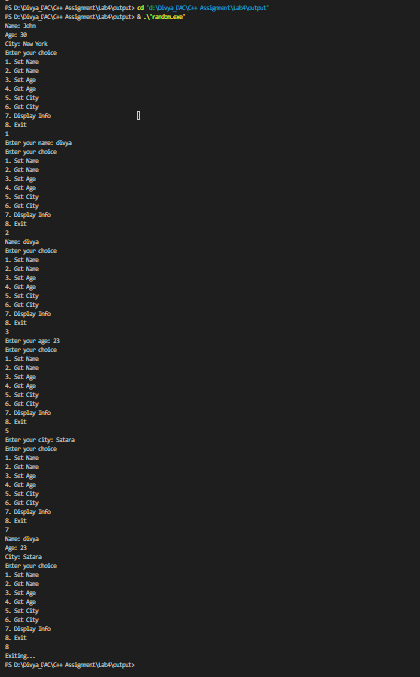
        }

    } while (choice != 8);

    return 0;

}

Output:



3. Create a class Date with data members as dd, mm, yy. Write getters and setters for all the data members. Also add the display function. Create Default and Parameterized constructors. Create the object of this class in main method and invoke all the methods in that class.

Program:

/\*Create a class Date with data members as dd, mm, yy.

Write getters and setters for all the data members. Also add the display function.

Create Default and Parameterized constructors. Create the

object of this class in main method and invoke all the methods in that class. \*/

#include<iostream>

using namespace std;

class Date{

    private:

    int dd;

    int mm;

    int yy;

    public:

    Date()

    {

        this->dd=0;

        this->mm=0;

        this->yy=0;

    }

    Date(int dd,int mm,int yy)

    {

        this->dd=dd;

        this->mm=mm;

        this->yy=yy;

    }

    void setDate(int dd)

    {

        this->dd=dd;

    }

    int getDate()

    {

        return this->dd;

    }

    void setMonth(int mm)

    {

        this->mm=mm;

    }

    int getMonth()

    {

        return this->mm;

    }

    void setYear(int yy)

    {

        this->yy=yy;

    }

    int getYear()

    {

        return this->yy;

    }

    void displayDate()

    {

        cout<<dd<<" / "<<mm<<" / "<<yy<<endl;

    }

};

int main()

{

    int dd,mm,yy;

    Date d1(14,6,2000);

    d1.displayDate();

    Date d2;

    cout<<"Enter Date: "<<endl;

    cin>>dd;

    d2.setDate(dd);

    cout<<"Set Date: "<<d2.getDate()<<endl;

    cout<<"Enter Month: "<<endl;

    cin>>mm;

    d2.setMonth(mm);

    cout<<"Set Date: "<<d2.getMonth()<<endl;

    cout<<"Enter Year: "<<endl;

    cin>>yy;

    d2.setYear(yy);

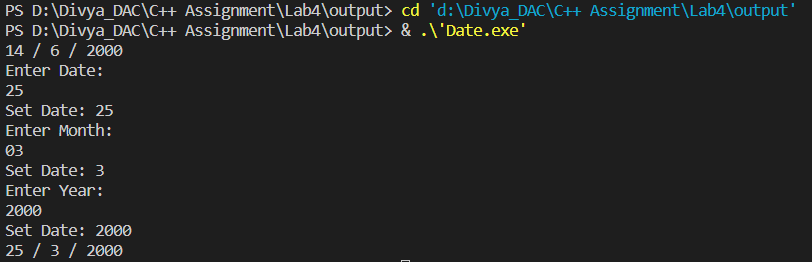
    cout<<"Set Date: "<<d2.getYear()<<endl;

    d2.displayDate();

    return 0;

}

Output:



4. Create a class Book with data members as bname,id,author,price. Write getters and setters for all the data members. Also add the display function. Create Default and Parameterized constructors. Create the object of this class in main method and invoke all the methods in that class.

Program:

/\*Create a class Book with data members as bname,id,author,price. Write getters and setters for all the

data members. Also add the display function. Create Default and Parameterized constructors. Create

the object of this class in main method and invoke all the methods in that class. \*/

#include<iostream>

using namespace std;

class Book{

    private:

    string bname;

    int id;

    string author;

    double price;

    public:

    Book()

    {

    bname="xyz";

    id=0;

    author="abc";

    price=0.00;

    }

    Book(string bname,int id,string author,double price)

    {

    this->bname=bname;

    this->id=id;

    this->author=author;

    this->price=price;

    }

    void displayInfo()

    {

        cout<<"Book Name: "<<bname<<endl;

        cout<<"Book ID: "<<id<<endl;

        cout<<"Book Author: "<<author<<endl;

        cout<<"Book Price: "<<price<<endl;

    }

    void setBname(string bname)

    {

        this->bname=bname;

    }

    string getBname()

    {

        return this->bname;

    }

    void setId(int id)

    {

        this->id=id;

    }

    int getId()

    {

        return this->id;

    }

    void setAuthor(string author)

    {

        this->author=author;

    }

    string getAuthor()

    {

        return this->author;

    }

    void setPrice(double price)

    {

        this->price=price;

    }

    double getPrice()

    {

        return this->price;

    }

};

int main()

{

    string bname;

    int id;

    string author;

    double price;

    Book b1("Reasoning",123,"S Chand",780);

    b1.displayInfo();

    Book b2;

    cout<<"Enter Book Name: "<<endl;

    getline(cin, bname);

    b2.setBname(bname);

    cout<<"Book Name: "<<b2.getBname()<<endl;

    cout<<"Enter Book Id: "<<endl;

    cin>>id;

    b2.setId(id);

    cout<<"Book Id: "<<b2.getId()<<endl;

    cout<<"Enter Book Author Name: "<<endl;

    //getline();

    fflush(stdin);

    getline(cin, author);

    b2.setAuthor(author);

    cout << "Book Author Name: " << b2.getAuthor() << endl;

    cout<<"Enter Book Price: "<<endl;

    cin>>price;

    b2.setPrice(price);

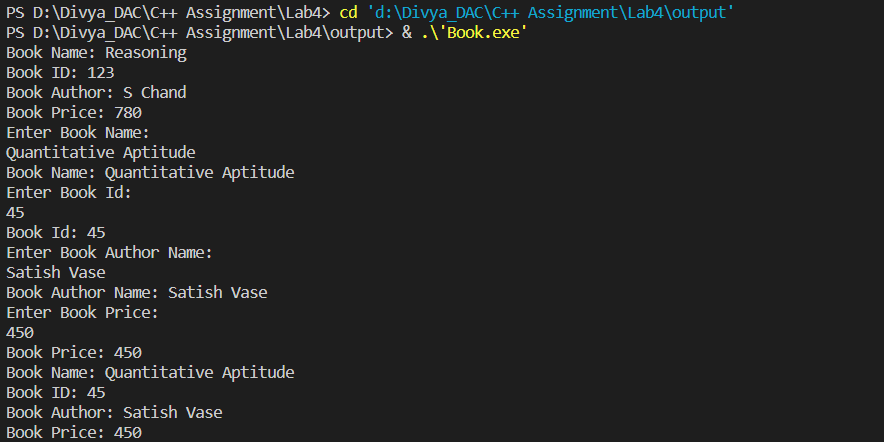
    cout<<"Book Price: "<<b2.getPrice()<<endl;

    b2.displayInfo();

    return 0;

}

Output:



5. Create a class Point with data members as x,y. Create Default and Parameterized constructors. Write getters and setters for all the data members. Also add the display function. Create the object of this class in main method and invoke all the methods in that class.

Program:

/\*Create a class Point with data members as x,y. Create Default and Parameterized constructors. Write

getters and setters for all the data members. Also add the display function. Create the object of this

class in main method and invoke all the methods in that class. \*/

#include<iostream>

using namespace std;

class Point{

    private:

    int x;

    int y;

    public:

    Point()

    {

      this->x=0;

      this->y=0;

    }

    Point(int x,int y)

    {

        this->x=x;

        this->y=y;

    }

    void setX(int x)

    {

        this->x=x;

    }

    int getX()

    {

        return this->x;

    }

    void setY(int Y)

    {

        this->y=Y;

    }

    int getY()

    {

        return this->y;

    }

    void displayPoints()

    {

        cout<<"X: "<<x<<endl;

        cout<<"Y: "<<y<<endl;

    }

};

int main()

{

    int x;

    int y;

    Point P1(10,20);

    P1.displayPoints();

    Point P2;

    cout<<"Enter Point X: "<<endl;

    cin>>x;

    P2.setX(x);

    cout<<"X: "<<x<<endl;

    cout<<"Enter Point Y: "<<endl;

    cin>>y;

    P2.setY(y);

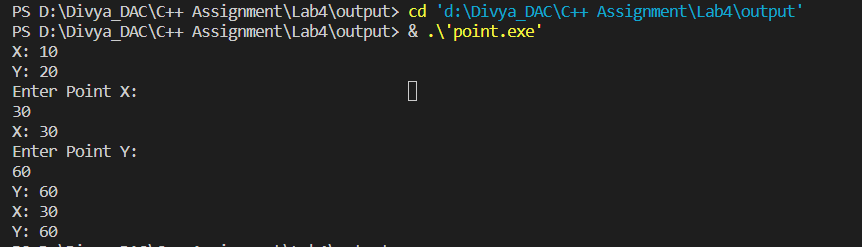
    cout<<"Y: "<<y<<endl;

    P2.displayPoints();

    return 0;

}

Output:



6. Create a class ComplexNumber with data members real, imaginary. Create Default and Parameterized constructors. Write getters and setters for all the data members. Also add the display function. Create the object of this class in main method and invoke all the methods in that class.

Program:

/\*Create a class ComplexNumber with data members real, imaginary.

Create Default and Parameterized constructors.

Write getters and setters for all the data members. Also add the display function.

Create the object of this class in main method and invoke all the methods in that class.\*/

#include<iostream>

using namespace std;

class ComplexNumber{

    private:

    int real;

    int imag;

    public:

    ComplexNumber()

    {

        this->real=0;

        this->imag=0;

    }

    ComplexNumber(int real,int imag)

    {

        this->real=real;

        this->imag=imag;

    }

    void setReal(int real)

    {

        this->real=real;

    }

    int getReal()

    {

        return this->real;

    }

    void setImag(int imag)

    {

        this->imag=imag;

    }

    int getImag()

    {

        return this->imag;

    }

    void displayComplex()

    {

        cout<<real<<" + "<<imag<<"j "<<endl;

    }

};

int main()

{

    int real;

    int imag;

    ComplexNumber c1(23,50);

    c1.displayComplex();

    ComplexNumber c2;

    cout<<"Enter Real Number: "<<endl;

    cin>>real;

    c2.setReal(real);

    cout<<"Enter Imag Number: "<<endl;

    cin>>imag;

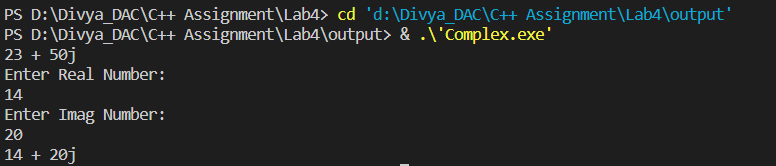
    c2.setImag(imag);

    c2.displayComplex();

    return 0;

}

Output:



**Lab5**



Animal Hierarchy:

Problem Statement: Create a hierarchy of animal classes. Start with a base class Animal and then create derived classes like Mammal, Bird, and Fish. Each of these derived classes should have specific properties and methods related to their respective categories of animals.

Program:

/\*Animal Hierarchy:

Problem Statement: Create a hierarchy of animal classes.

Start with a base class Animal and then

create derived classes like Mammal, Bird, and Fish.

 Each of these derived classes should have specific properties

 and methods related to their respective categories of animals.

\*/

#include<iostream>

using namespace std;

class Animal{

    private:

    string name;

    int leg;

    string survive;

    public:

    Animal()

    {

        cout<<"---------Inside def const of Animal----------"<<endl;

        name="default";

        leg=0;

        survive="Earth";

    }

    Animal(string name,int leg,string survive)

    {

        cout<<"---------Inside prameterised const of Animal----------"<<endl;

        this->name=name;

        this->leg=leg;

        this->survive=survive;

    }

    void displayInfo()

    {

        cout<<"Name: "<<this->name<<endl;

        cout<<"No.of legs: "<<this->leg<<endl;

        cout<<"Survival info: "<<this->survive<<endl;

    }

};

class Mammal:public Animal

{

    private:

       string food;

    public:

       Mammal()

       {

        cout<<"-----Inside def const of Mammels class-------"<<endl;

        food="Green grass";

       }

       Mammal(string name,int leg,string survive,string food):Animal(name,leg,survive)

       {

        cout<<"-----Inside parameterised const of Mammels class-------"<<endl;

        this->food= food;

       }

       void Walking()

       {

        cout<<"Mammels are terrestial animals"<<endl;

       }

       void displayFood()

       {

        cout<<"Food: "<<this->food<<endl;

       }

};

class Fish:public Animal

{

    private:

       string food;

    public:

       Fish()

       {

        cout<<"-----Inside def const of Fish class-------"<<endl;

        food="default";

       }

       Fish(string name,int leg,string survive,string food):Animal(name,leg,survive)

       {

        cout<<"-----Inside parameterised const of Fish class-------"<<endl;

       this->food=food;

       }

       void Swimming()

       {

        cout<<"Fish can swim"<<endl;

       }

       void displayFood()

       {

        cout<<"Food: "<<this->food<<endl;

       }

};

class Bird:public Animal

{

    private:

       string food;

    public:

       Bird()

       {

        cout<<"-----Inside def const of Mammels class-------"<<endl;

        food="Green grass";

       }

       Bird(string name,int leg,string survive,string food):Animal(name,leg,survive)

       {

        cout<<"-----Inside parameterised const of Mammels class-------"<<endl;

        this->food=food;

       }

       void Flying()

       {

        cout<<"Birds can fly"<<endl;

       }

       void displayFood()

       {

        cout<<"Food: "<<food;

       }

};

int main()

{

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Animal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

     Animal a1;

     a1.displayInfo();

    Animal a2("Horse",4,"stable");

     a2.displayInfo();

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Mammmal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    Mammal m1;

    m1.displayInfo();

    Mammal m2("Human",2,"House","fast food");

    m2.displayInfo();

    m2. Walking();

    m2.displayFood();

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fish\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    Fish f1;

    f1.displayInfo();

    Fish f2("Dolphine",0,"Sea","Seafood");

    f2.displayInfo();

    f2.Swimming();

    f2.displayFood();

    Bird b1;

    b1.displayInfo();

    Bird b2("Parrot",2,"tree","guava");

    b2.displayInfo();

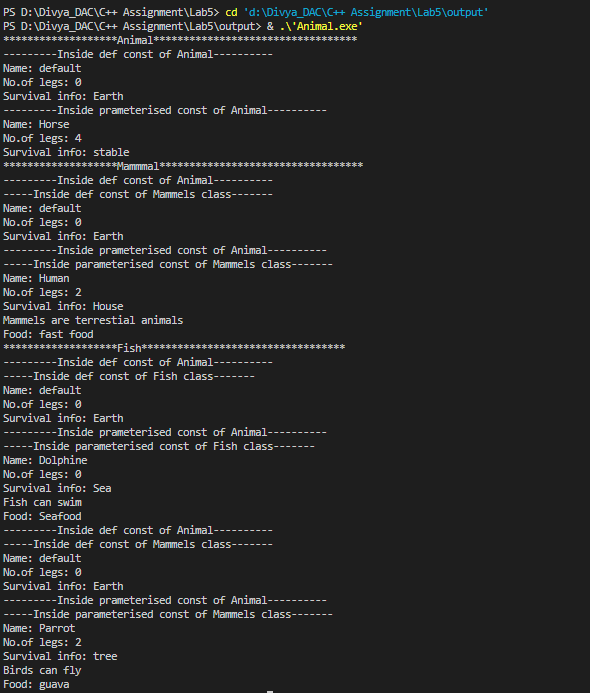
    b2.Flying();

    b2.displayFood();

    return 0;

}

Output:



Shape Hierarchy:

Problem Statement: Design a hierarchy of shape classes. Begin with a base class Shape and then create derived classes like Circle, Rectangle, and Triangle. Each shape should have methods for calculating area and perimeter specific to its geometry.

Program:

/\*Shape Hierarchy:

Problem Statement: Design a hierarchy of shape classes.

 Begin with a base class Shape and then create derived classes like Circle, Rectangle, and Triangle.

Each shape should have methods for calculating area and perimeter specific to its geometry.\*/

#include<iostream>

using namespace std;

class shape{

   int vertices;

   int sides ;

 public:

   shape(){

    vertices = 0;

    sides = 0 ;

    cout<<"-------In shape default Constructor ;"<<endl;

   }

   shape(int vertices , int sides ){

     this->vertices= vertices;

     this->sides=sides;

     cout<<"----------In shape parametrized constructor"<<endl;

   }

   virtual void displayParameters(){

      cout<<"Vertices are :"<<vertices<<endl;

      cout<<"sides are :"<<sides<<endl;

   } };

   class Circle : public shape {

    float radius ;

   public:

    Circle(){

       radius = 5;

       cout<<"------Inside Circle default Constructor---------"<<endl;

    }

    Circle (int vertices , int sides,int radius): shape( vertices ,  sides ){

       this->radius=radius;

       cout<<"------Inside Circle parameterised Constructor---------"<<endl;

    }

    void displayParameters()

    {

      shape::displayParameters();

      cout<<"Radius: "<<radius<<endl;

    }

    void CircleArea(){

        float area = 3.14 \* radius \* radius ;

        cout<<"Area of circle is :"<<area<<endl;

        float perimeter = 2 \* 3.14 \* radius;

        cout<<"Perimeter of Circle is :"<<perimeter<<endl;

    }

   };

    class Triangle : public shape {

    float height  ;

    float base ;

   public:

   Triangle (){

        height = 0 ;

     base =0;

       cout<<"------Inside Triangle default Constructor---------"<<endl;

    }

    Triangle (int vertices , int sides, float base , float height): shape( vertices ,  sides ){

       this->height= height;

       this->base = base ;

       cout<<"------Inside Triangle parameterised Constructor---------"<<endl;

    }

     void displayParameters()

    {

      shape::displayParameters();

      cout<<"Base: "<<base<<endl;

      cout<<"Height: "<<height<<endl;

    }

    void TriangleArea(){

        float area1 = 0.5\*base\*height;

        cout<<"Area of Triangle is :"<<area1<<endl;

        float perimeter = base\*3;

        cout<<"Perimeter of Triangle is :"<<perimeter<<endl;

    }

   };

class Rectangle : public shape {

    float breadth  ;

    float length ;

   public:

   Rectangle(){

        breadth =0 ;

       length =0;

       cout<<"------Inside Rectangle default Constructor---------"<<endl;

    }

    Rectangle (int vertices , int sides, float breadth , float length): shape( vertices ,  sides ){

       this->length= length;

       this->breadth = breadth ;

       cout<<"------Inside Rectangle Parameterised Constructor---------"<<endl;

    }

     void displayParameters()

    {

      shape::displayParameters();

      cout<<"Length: "<<length<<endl;

      cout<<"Breadth: "<<breadth<<endl;

    }

    void RectangleArea(){

        float area1 = length\*breadth;

        cout<<"Area of Rectangle is :"<<area1<<endl;

        float perimeter = 2\*(length\*breadth);

        cout<<"Perimeter of rectangle is :"<<perimeter<<endl;

    }

   };

int main(){

   Rectangle R1(4,4,20,25);

    R1.displayParameters();

   R1.RectangleArea();

   Circle C1(1, 1, 10);

   C1.displayParameters();

   C1.CircleArea();

    Triangle T1(3, 3, 10, 15);

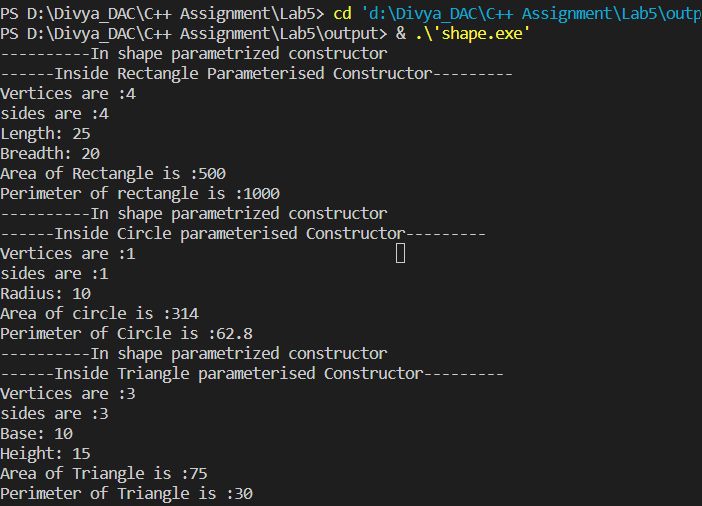
    T1.displayParameters();

    T1.TriangleArea();

    return 0 ;

}

Output:



Employee Inheritance:

Problem Statement: Build a system for managing employees. Create a base class Employee with attributes such as name, employee ID, and salary. Then, derive classes like Manager and Developer, each with its own attributes and methods. Implement a common method, like calculate\_salary(), in the base class.

Program:

/\*Problem Statement: Build a system for managing employees.

Create a base class Employee with attributes such as name, employee ID, and salary.

Then, derive classes like Manager and Developer, each with its own attributes and methods.

Implement a common method, like calculate\_salary(), in the base class.\*/

#include<iostream>

using namespace std;

class Employee{

    private:

    string name;

    int id;

    double sal;

    public:

    Employee( string name,int id,double sal)

    {

        this->name=name;

        this->id=id;

        this->sal=sal;

    }

    virtual void displayInfo()

    {

        cout<<name<<endl;

        cout<<id<<endl;

        cout<<sal<<endl;

    }

    virtual void calSal(){

        cout<<"Salary: "<<sal<<endl;

    }

};

class Manager:public Employee{

    private:

    string tName;

    public:

    Manager(string name,int id,double sal,string tName):Employee(name,id,sal)

    {

       this->tName=tName;

    }

    void isManaging()

    {

        cout<<"Manager is managing a team"<<endl;

    }

    void displayInfo()

    {

        Employee::displayInfo();

        cout<<tName<<endl;

    }

};

class Developer:public Employee{

    private:

    string specialisation;

    public:

    Developer(string name,int id,double sal,string specialisation):Employee(name,id,sal)

    {

       this->specialisation=specialisation;

    }

    void isDebugging()

    {

        cout<<"Developer is debugging a code"<<endl;

    }

    void isCoding()

    {

        cout<<"Developer is coding"<<endl;

    }

    void displayInfo()

    {

        Employee::displayInfo();

        cout<<specialisation<<endl;

    }

};

int main()

{

    Manager M1("Mr.Gupta",123,80000,"ITDept");

    M1.displayInfo();

    M1.isManaging();

    M1.calSal();

    Developer D1("Supriya",456,60000,"FullStack");

    D1.displayInfo();

    D1.isCoding();

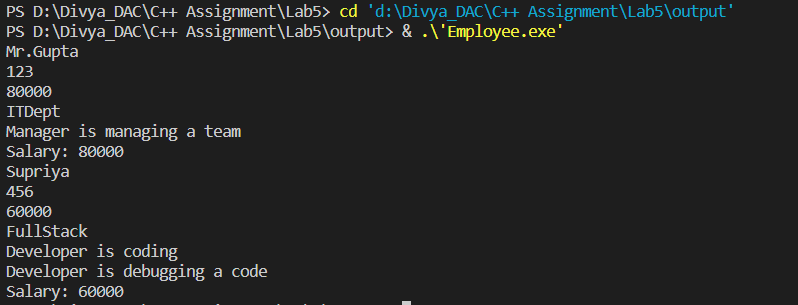
    D1.isDebugging();

    D1.calSal();

    return 0;

}

Output:



Vehicle Inheritance:

Problem Statement: Develop a class hierarchy for vehicles. Start with a base class Vehicle and create derived classes like Car, Motorcycle, and Truck. Each derived class should have unique properties like the number of wheels and specific methods like start\_engine().

Program:

/\*Develop a class hierarchy for vehicles.

Start with a base class Vehicle and create derived classes like Car, Motorcycle, and Truck.

Each derived class should have unique properties like the number of wheels

and specific methods like start\_engine().\*/

#include<iostream>

using namespace std;

class Vehicle{

    private :

    string name;

    int speed;

    public:

    Vehicle(string name="\0",int speed=0 )

    {

        this->name=name;

        this->speed=speed;

    }

   virtual void displayInfo()

    {

        cout<<"Name: "<<name<<endl;

        cout<<"Speed: "<<speed<<endl;

    }

};

class Car: public Vehicle{

    private:

    int noOfWheels;

    string colour;

    public:

    Car(string name,int speed,int noOfWheels=0,string colour="\0"):Vehicle(name,speed)

    {

        this->noOfWheels=noOfWheels;

        this->colour=colour;

    }

    void start\_engine()

    {

        cout<<"Car engine is strated"<<endl;

    }

    void displayInfo()

    {

        Vehicle::displayInfo();

        cout<<"No Of Wheels: "<<noOfWheels<<endl;

        cout<<"Car color: "<<colour<<endl;

    }

};

class Motorcycle: public Vehicle{

    private:

    int noOfWheels;

    string colour;

    public:

    Motorcycle(string name,int speed,int noOfWheels=0,string colour="\0"):Vehicle(name,speed)

    {

        this->noOfWheels=noOfWheels;

        this->colour=colour;

    }

    void start\_engine()

    {

        cout<<"Motorcycle engine is strated"<<endl;

    }

    void displayInfo()

    {

         Vehicle::displayInfo();

        cout<<"No Of Wheels: "<<noOfWheels<<endl;

        cout<<"Motorcycle color: "<<colour<<endl;

    }

};

class Truck: public Vehicle{

    private:

    int noOfWheels;

    string colour;

    public:

    Truck(string name,int speed,int noOfWheels=0,string colour="\0"):Vehicle(name,speed)

    {

        this->noOfWheels=noOfWheels;

        this->colour=colour;

    }

    void start\_engine()

    {

        cout<<"Truck engine is strated"<<endl;

    }

    void displayInfo()

    {

        Vehicle::displayInfo();

        cout<<"No Of Wheels: "<<noOfWheels<<endl;

        cout<<"Truck color: "<<colour<<endl;

    }

};

int main()

{

    Car c("Wagnor",100,4,"NavyBlue");

    c.displayInfo();

    c.start\_engine();

    Motorcycle m("Honda",50,2,"Silver");

    m.displayInfo();

    m.start\_engine();

    Truck t("Bajaj",120,8,"Red");

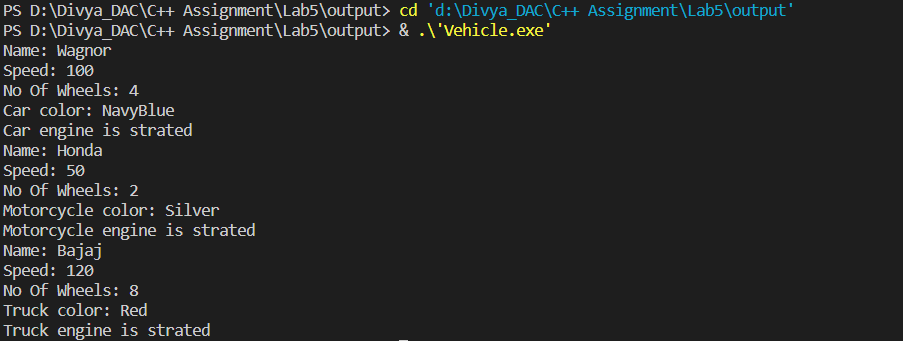
    t.displayInfo();

    t.start\_engine();

    return 0;

}

Output:



Bank Account Inheritance:

Problem Statement: Design a system for managing bank accounts. Create a base class BankAccount with attributes like account number and balance. Derive classes like SavingsAccount and CheckingAccount, each with specialized methods like withdraw() and calculate\_interest().

Program:

/\*Bank Account Inheritance:

Problem Statement: Design a system for managing bank accounts.

 Create a base class BankAccount with attributes like account number and balance.

  Derive classes like SavingsAccount and CheckingAccount,

  each with specialized methods like withdraw() and calculate\_interest().

\*/

#include<iostream>

using namespace std;

class BankAccount{

 int accountNumber;

   protected : int Initbalance;

 public:

 BankAccount(int accountNumber = 0, int Initbalance=0){

    this->accountNumber=accountNumber;

    this->Initbalance=Initbalance;

 }

  virtual void displayAccDetails(){

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

    cout<<"balance :"<<Initbalance<<endl;

}

};

class SavingsAccount : public BankAccount

{

    int CurrentBalance ;

public :

SavingsAccount(int accountNumber, int Initbalance , int CurrentBalance ) : BankAccount(accountNumber , Initbalance){

  this->CurrentBalance=CurrentBalance ;

}

void withdraw(int amount){

      if(amount>CurrentBalance)

      cout<<"You have insufficient balance"<<endl;

      else

      CurrentBalance=CurrentBalance-amount;

}

void displayAccDetails(){

    BankAccount::displayAccDetails();

    cout<<"Current Balance :"<<CurrentBalance<<endl;

}

};

class CheckingAccount : public BankAccount

{

     private :

     int n ;

     int Rate = 10;

public :

CheckingAccount(int accountNumber, int Initbalance , int n  ): BankAccount(accountNumber , Initbalance)

{

 this->n=n;

}

void CalIneterest(){

     double  Interest = Initbalance\*Rate\*n\*0.01;

    cout<<"Interest is :"<<Interest<<endl;

}

};

int main(){

     SavingsAccount s (21254 ,500000,500000);

     s.withdraw(100000);

     s.displayAccDetails();

     cout<<"---------------------------------------------"<<endl;

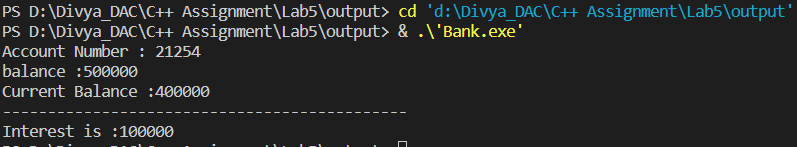
     CheckingAccount c (21589,200000,5);

     c.CalIneterest();

     return  0 ;

}

Output:



Geometric Shapes with Polymorphism:

Problem Statement: Extend the shape hierarchy example by implementing polymorphism. Define a base class Shape with methods to calculate area and perimeter. Then, create derived classes like Circle, Rectangle, and Triangle, each with its own implementation of these methods.

Program:

/\*Geometric Shapes with Polymorphism:

Problem Statement: Extend the shape hierarchy example by implementing polymorphism.

Define a base class Shape with methods to calculate area and perimeter.

Then, create derived classes like Circle, Rectangle, and

Triangle, each with its own implementation of these methods.

\*/

#include<iostream>

using namespace std;

class Shape{

    public:

    virtual void calculateArea()=0;

    virtual void calculatePerimeter()=0;

};

class Circle:public Shape{

    private:

    float r;

    public:

    Circle()

    {

        this->r=0.0;

    }

    Circle(float r)

    {

        this->r=r;

    }

    void calculateArea()

    {

        cout<<"Circle Area: "<<(3.14\*r\*r)<<endl;

    }

    void calculatePerimeter()

    {

        cout<<"Circle Perimeter: "<<(2\*3.14\*r)<<endl;

    }

};

class Rectangle:public Shape{

    private:

    float length;

    float width;

    public:

    Rectangle()

    {

        this->length=0.0;

        this->width=0.0;

    }

    Rectangle(float length,float width)

    {

        this->length=length;

        this->width=width;

    }

    void calculateArea()

    {

        cout<<"Rectangle Area: "<<(length\*width)<<endl;

    }

    void calculatePerimeter()

    {

        cout<<"Rectangle Perimeter: "<<((length+width)/2)<<endl;

    }

};

class Triangle:public Shape{

    private:

    float height;

    float base;

    float side1;

    float side2;

    float side3;

    public:

    Triangle()

    {

        this->height=0.0;

        this->base=0.0;

    }

    Triangle(float height,float base,float side1,float side2,float side3)

    {

        this->height=height;

        this->base=base;

        this->side1=side1;

        this->side2=side2;

        this->side3=side3;

    }

    void calculateArea()

    {

        cout<<"Triangle Area: "<<((base\*height)/2)<<endl;

    }

    void calculatePerimeter()

    {

        cout<<"Triangle Perimeter: "<<(side1+side2+side3)<<endl;

    }

};

int main()

{

    Circle c(5.0);

    c.calculateArea();

    c.calculatePerimeter();

    Rectangle r1(15.0,10.0);

    r1.calculateArea();

    r1.calculatePerimeter();

    Triangle t1(15.0,10.0,2.0,3.0,4.0);

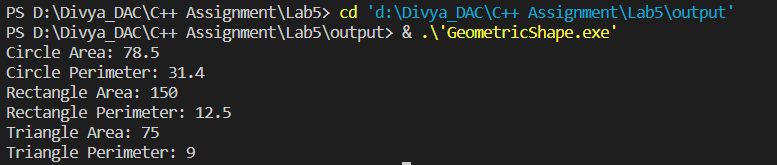
    t1.calculateArea();

    t1.calculatePerimeter();

    return 0;

}

Output:



Person and Student Inheritance:

Problem Statement: Model a system for handling individuals and students within an educational institution. Create a base class Person with attributes like name and age. Derive a Student class with additional attributes like student ID and GPA, inheriting the common attributes from the Person class.

Program:

/\*Person and Student Inheritance:

Problem Statement: Model a system for handling individuals and students

within an educational institution. Create a base class Person with attributes like name and age.

Derive a Student class with additional attributes like student ID and CGPA,

inheriting the common attributes from the Person class.

\*/

#include<iostream>

using namespace std;

class Person{

    private:

    string name;

    int age;

    public:

    Person(string name,int age )

    {

        this->name=name;

        this->age=age;

    }

    virtual void displayInfo()

    {

        cout<<"Name: "<<name<<endl;

         cout<<"Age: "<<age<<endl;

    }

};

class Student:public Person{

    private:

    int ID;

    float CGPA;

    public:

    Student(string name,int age,int ID,float CGPA ):Person(name,age)

    {

        this->ID=ID;

        this->CGPA=CGPA;

    }

    void displayInfo()

    {

        Person::displayInfo();

        cout<<"Student ID: "<<ID<<endl;

         cout<<"CGPA: "<<CGPA<<endl;

    }

};

class Teacher:public Person{

    private:

    int ID;

    string dept;

    public:

    Teacher(string name,int age,int ID,string dept ):Person(name,age)

    {

        this->ID=ID;

        this->dept=dept;

    }

    void displayInfo()

    {

        Person::displayInfo();

        cout<<"Teacher ID: "<<ID<<endl;

         cout<<"Department Name: "<<dept<<endl;

    }

};

int main()

{

    Student s1("Anchal",23,122,9.95);

    s1.displayInfo();

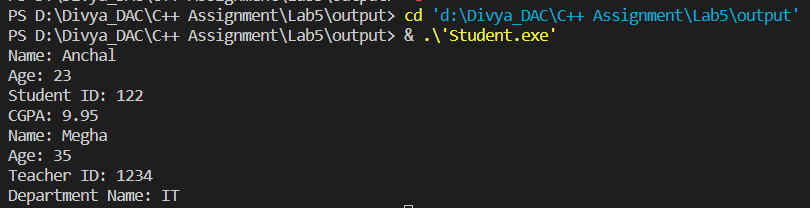
    Teacher t1("Megha",35,1234,"IT");

    t1.displayInfo();

    return 0;

}

Output:



Library Catalog with Books and Journals:

Problem Statement: Build a library catalog system. Create a base class LibraryItem with properties like title and author. Then, derive classes like Book and Journal, each with their unique properties. Implement methods to check out and return items in the derived classes.

Program:

/\*Library Catalog with Books and Journals:

Problem Statement: Build a library catalog system .

Create a base class LibraryItem with properties like title and author.

Then, derive classes like Book and Journal, each with their unique properties.

Implement methods to check out and return items in the derived classes.

\*/

#include<iostream>

using namespace std;

class LibraryItem{

    string title ;

    string author ;

public:

    LibraryItem( string title="abc" ,string author ="xyz" ){

    this->title = title ;

    this->author=author;

    }

    virtual void display(){

        cout<<"Title of Book :"<<title<<endl;

        cout<<"Author of Book:"<<author<<endl;

    }

};

class Book : public LibraryItem{

  int price ;

  int bookId ;

 public:

  Book(string title , string author , int price , int bookId):LibraryItem( title ,author){

     this->bookId=bookId;

     this->price=price;

  }

  void display(){

        LibraryItem::display();

        cout<<"Book Id :"<<bookId<<endl;

        cout<<"price :"<<price<<endl;

    }

};

class Journal : public LibraryItem{

  int edition ;

 int  price ;

public:

  Journal(string title  , string author , int price , int edition):LibraryItem( title ,author){

     this->edition=edition;

     this->price=price;

  }

  void display(){

        LibraryItem::display();

        cout<<"edition is  :"<<edition<<endl;

        cout<<"price :"<<price<<endl;

    }

};

int main ()

{

     Book b ("Concept of Physics","HC Verma",800,121);

        b.display();

    Journal j ("Antarang" , "Anant Sharma", 500,10)  ;

        j.display();

    cout<<"------------------------------------"<<endl;

       int choice;

       string title;

       string author;

       int price;

       int bookId;

       int edition;

       int i=0;

       LibraryItem\*lb[5];

       int count=0;

      do{

        cout<<"Enter your choice: \n1.Book\n2.Journal\n3.exit"<<endl;

       cin >> choice;

           switch(choice)

           {

            case 1:

            cout<<"enter title "<<endl;

            fflush(stdin);

            getline(cin,title);

           cout<<"enter author "<<endl;

           fflush(stdin);

            getline(cin,author);

           cout<<"enter price "<<endl;

           cin>>price;

           cout<<"enter bookId "<<endl;

           cin>>bookId;

          lb[i++]=new Book(title,author,price,bookId);

          break;

          case 2:

            cout<<"enter title "<<endl;

            fflush(stdin);

            getline(cin,title);

            cout<<"enter author "<<endl;

            fflush(stdin);

            getline(cin,author);

           cout<<"enter price "<<endl;

           cin>>price;

           cout<<"enter edition "<<endl;

           cin>>edition;

          lb[i++]=new Journal(title,author,price,edition);

          break;

          }

      }while (choice!=3);

       for(int i=0;i<3;i++)

       {

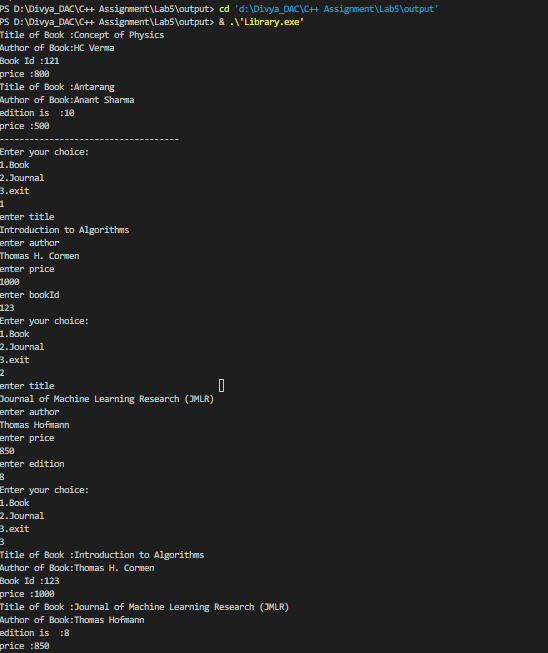
        lb[i]->display();

       }

    return  0 ;

}

Output:



Shape Sorting with Interfaces:

Problem Statement: Implement a shape sorting program. Define a base class Shape with properties like area and perimeter. Create derived classes like Circle, Rectangle, and Triangle. Implement an interface Sortable with a method to compare shapes by area. Use this interface to sort a list of shapes.

Program:

/\*Shape Sorting with Interfaces:

Problem Statement: Implement a shape sorting program.

Define a base class Shape with properties like area and perimeter.

Create derived classes like Circle, Rectangle, and Triangle.

Implement an interface Sortable with a method to compare shapes by area. Use this interface to sort a list of shapes.

\*/

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

class Sortable {

public:

    virtual void sortArea(vector<float>& arr) = 0;

};

class Shape {

public:

    virtual float calArea() = 0;

};

class Circle : public Shape {

private:

    float r;

public:

    Circle(float r) : r(r) {}

    float calArea() override {

        return 3.14 \* r \* r;

    }

};

class Rectangle : public Shape {

private:

    float l, b;

public:

    Rectangle(float l, float b) : l(l), b(b) {}

    float calArea() override {

        return l \* b;

    }

};

class Triangle : public Shape {

private:

    float b, h;

public:

    Triangle(float b, float h) : b(b), h(h) {}

    float calArea() override {

        return (b \* h) / 2;

    }

};

class ShapeSorter : public Sortable {

public:

    void sortArea(vector<float>& arr) override {

        sort(arr.begin(), arr.end());

    }

};

int main() {

    vector<float> arr\_circle;

    vector<float> arr\_rectangle;

    vector<float> arr\_triangle;

    Circle circles[] = { Circle(10.0), Circle(20.0), Circle(30.0) };

    Rectangle rectangles[] = { Rectangle(5.0, 10.0), Rectangle(3.0, 6.0), Rectangle(8.0, 12.0) };

    Triangle triangles[] = { Triangle(5.0, 10.0), Triangle(3.0, 6.0), Triangle(8.0, 12.0) };

    for (auto& circle : circles)

        arr\_circle.push\_back(circle.calArea());

    for (auto& rectangle : rectangles)

        arr\_rectangle.push\_back(rectangle.calArea());

    for (auto& triangle : triangles)

        arr\_triangle.push\_back(triangle.calArea());

    ShapeSorter sorter;

    cout << "Sorting circles by area:" << endl;

    sorter.sortArea(arr\_circle);

    for (const auto& area : arr\_circle)

        cout << area << endl;

    cout << "Sorting rectangles by area:" << endl;

    sorter.sortArea(arr\_rectangle);

    for (const auto& area : arr\_rectangle)

        cout << area << endl;

    cout << "Sorting triangles by area:" << endl;

    sorter.sortArea(arr\_triangle);

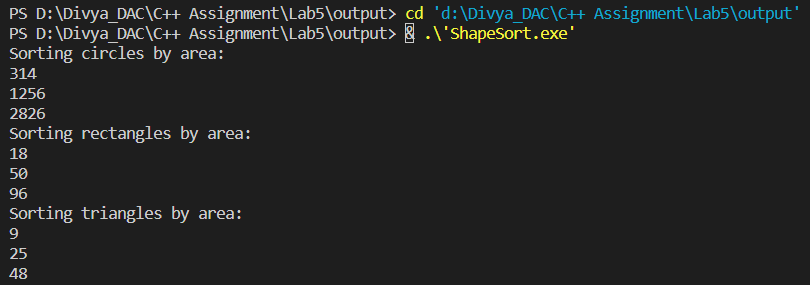
    for (const auto& area : arr\_triangle)

        cout << area << endl;

    return 0;

}

Output:



Employee Payroll System with Abstract Classes:

Problem Statement: Design an employee payroll system. Create an abstract class Employee with attributes like name and employee ID. Derive concrete classes like HourlyEmployee and SalariedEmployee. Define abstract methods for calculating pay in the base class and implement them in the derived classes.

Program:

/\*Employee Payroll System with Abstract Classes:

Problem Statement: Design an employee payroll system.

Create an abstract class Employee with attributes like name and employee ID.

 Derive concrete classes like HourlyEmployee and SalariedEmployee.

 Define abstract methods for calculating pay in the base class and implement them in the derived classes.\*/

 #include<iostream>

 using namespace std;

 class Employee{

    private :

    string name;

    int ID;

    public:

    Employee(string name,int ID)

    {

        this->name=name;

        this->ID=ID;

    }

    virtual void calculatePay()=0;

 };

 class HourlyEmployee: public Employee

 {

    private:

    int hrsWorked;

    int rate=500;

    public:

    HourlyEmployee(string name,int id,int hrsWorked):Employee(name,id)

    {

        this->hrsWorked=hrsWorked;

    }

    void calculatePay()

    {

        cout<<"Hourly Employee Salary: "<<(hrsWorked\*rate)<<endl;

    }

 };

 class SalariedEmployee: public Employee

 {

    private:

    double basicSalary=20000;

    double bonus=10000;

    ;

    public:

    SalariedEmployee(string name,int id):Employee(name,id)

    {

    }

    void calculatePay()

    {

        cout<<"Salaried Employee Salary: "<<(basicSalary+bonus)<<endl;

    }

 };

 int main()

 {

    HourlyEmployee E("Tejas",123,120);

    E.calculatePay();

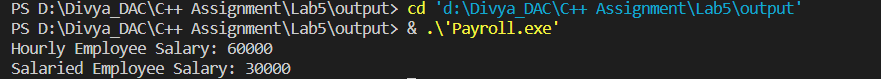
    SalariedEmployee S("Rahul",123);

    S.calculatePay();

    return 0;

 }

Output:



**Lab6**



1 Solve this.

Fresh business scenario to apply inheritance , polymorphism to emp based organization scenario.

Create Emp based organization structure --- Emp , Mgr , Worker

1.1 Emp state--- id(int), name, deptId , basicSalary(double)

Accept all of above in constructor arguments.

Methods ---

1.2. compute net salary ---ret 0

(eg : public double computeNetSalary(){return 0;})

1.2 Mgr state ---id,name,basic,deptId , perfBonus

Add suitable constructor

Methods ----

1. compute net salary (formula: basic+perfBonus) -- override computeNetSalary

1.3 Worker state --id,name,basic,deptId,hoursWorked,hourlyRate

Methods :

1. compute net salary (formula: = basic+(hoursWorked\*hourlyRate) --override computeNetSalary

2. get hrlyRate of the worker -- add a new method to return hourly rate of a worker.(getter)

Create suitable array to store organization details.

Provide following options

1. Hire Manager

I/P : all manager details

2. Hire Worker

I/P : all worker details

3. Display information of all employees net salary (by invoking computeNetSal),

4. Exit

Program:

/\*Solve this.

Fresh business scenario to apply inheritance , polymorphism   to emp based organization scenario.

Create Emp based organization structure --- Emp , Mgr , Worker

1.1 Emp state--- id(int), name, deptId , basicSalary(double)

Accept all of above in constructor arguments.

Methods ---

1.2. compute net salary ---ret 0

(eg : public double computeNetSalary(){return 0;})

1.2 Mgr state  ---id,name,basic,deptId , perfBonus

Add suitable constructor

Methods ----

1. compute net salary (formula: basic+perfBonus) -- override computeNetSalary

1.3 Worker state  --id,name,basic,deptId,hoursWorked,hourlyRate

Methods :

1.  compute net salary (formula:  = basic+(hoursWorked\*hourlyRate) --override computeNetSalary

2. get hrlyRate of the worker  -- add a new method to return hourly rate of a worker.(getter)

Create suitable array to store organization details.

Provide following options

1. Hire Manager

I/P : all manager details

2. Hire Worker

I/P : all worker details

3. Display information of all employees net salary (by invoking computeNetSal),

4. Exit\*/

#include<iostream>

#include<string.h>

using namespace std;

class Employee{

    private:

      int id;

      string name;

      int deptId ;

     protected:

     double basicSalary;

     public:

     Employee()

     {

        id=0;

        name="\0";

        deptId=0;

        basicSalary=0.00;

     }

     Employee(int id,string name,int deptId,double basicSalary)

     {

        this->id=id;

        this->name=name;

        this->deptId=deptId;

        this->basicSalary=basicSalary;

     }

     virtual double computeNetSalary()

     {

        return basicSalary;

     }

     virtual void display()

    {

            cout<<"EmpID: "<<id<<"   "<<"EmpName: "<<name<<"    "<<"DeptID: "<<deptId<<"   "<<"Basic Salary: "<<basicSalary<<endl;

    }

};

class Mgr:public Employee{

    private:

   double perfBonus;

   public:

   Mgr()

   {

    perfBonus=0;

   }

   Mgr(int id,string name,int deptId,double basicSalary,double perfBonus):Employee(id,name,deptId,basicSalary)

   {

    this->perfBonus=perfBonus;

   }

   double computeNetSalary()

    {

        return (basicSalary + perfBonus);

    }

    void display()

    {

        Employee::display();

        cout<<"PerfBonus: "<<perfBonus<<endl;

    }

};

class Worker:public Employee{

    private:

    int hoursWorked;

    int hourlyRate;

    public:

    Worker()

    {

            hoursWorked=0;

            hourlyRate=0;

    }

    Worker(int id,string name,int deptId,double basicSalary,int hoursWorked,int hourlyRate):Employee(id,name,deptId,basicSalary)

    {

        this->hoursWorked=hoursWorked;

        this->hourlyRate=hourlyRate;

    }

     int getHourlyRate()

    {

        return this->hourlyRate;

    }

    double computeNetSalary()

    {

        return basicSalary+(hoursWorked\*hourlyRate);

    }

    void display()

    {

        Employee::display();

        cout<<" hoursWorked: "<< hoursWorked<<"hourlyRate: "<<hourlyRate<<endl;

    }

};

int main()

{

        int id;

        string name;

        int deptId;

        double basicSalary;

        double perfBonus;

            cout<<"Enter Manager 1 details: "<<endl;

            cout<<"ENter ID:"<<endl;

            cin>>id;

            cout<<"Enter Name:"<<endl;

            cin>>name;

            cout<<"ENter deptID:"<<endl;

            cin>>deptId;

            cout<<"ENter basicSalary:"<<endl;

            cin>>basicSalary;

            cout<<"ENter perfBonus:"<<endl;

            cin>>perfBonus;

            Mgr m1(id,name,deptId,basicSalary,perfBonus);

       cout<<"Enter Manager 2 details: "<<endl;

            cout<<"ENter ID:"<<endl;

            cin>>id;

            cout<<"Enter Name:"<<endl;

            cin>>name;

            cout<<"ENter deptID:"<<endl;

            cin>>deptId;

            cout<<"ENter basicSalary:"<<endl;

            cin>>basicSalary;

            cout<<"ENter perfBonus:"<<endl;

            cin>>perfBonus;

            Mgr m2(id,name,deptId,basicSalary,perfBonus);

        cout<<"Enter Manager 3 details: "<<endl;

            cout<<"ENter ID:"<<endl;

            cin>>id;

            cout<<"Enter Name:"<<endl;

            cin>>name;

            cout<<"ENter deptID:"<<endl;

            cin>>deptId;

            cout<<"ENter basicSalary:"<<endl;

            cin>>basicSalary;

            cout<<"ENter perfBonus:"<<endl;

            cin>>perfBonus;

            Mgr m3(id,name,deptId,basicSalary,perfBonus);

            int hoursWorked;

            int hourlyRate;

         cout<<"Enter Worker 1 details: "<<endl;

            cout<<"ENter ID:"<<endl;

            cin>>id;

            cout<<"Enter Name:"<<endl;

            cin>>name;

            cout<<"ENter deptID:"<<endl;

            cin>>deptId;

            cout<<"ENter basicSalary:"<<endl;

            cin>>basicSalary;

            cout<<"ENter hoursWorked:"<<endl;

            cin>>hoursWorked;

            cout<<"hourlyRate:"<<endl;

            cin>>hourlyRate;

            Worker wr1(id,name,deptId,basicSalary,hoursWorked,hourlyRate);

   cout<<"Enter Worker 2 details: "<<endl;

            cout<<"ENter ID:"<<endl;

            cin>>id;

            cout<<"Enter Name:"<<endl;

            cin>>name;

            cout<<"ENter deptID:"<<endl;

            cin>>deptId;

            cout<<"ENter basicSalary:"<<endl;

            cin>>basicSalary;

            cout<<"ENter hoursWorked:"<<endl;

            cin>>hoursWorked;

            cout<<"hourlyRate:"<<endl;

            cin>>hourlyRate;

            Worker wr2(id,name,deptId,basicSalary,hoursWorked,hourlyRate);

 cout<<"Enter Worker 3 details: "<<endl;

           cout<<"ENter ID:"<<endl;

            cin>>id;

            cout<<"Enter Name:"<<endl;

            cin>>name;

            cout<<"ENter deptID:"<<endl;

            cin>>deptId;

            cout<<"ENter basicSalary:"<<endl;

            cin>>basicSalary;

            cout<<"ENter hoursWorked:"<<endl;

            cin>>hoursWorked;

            cout<<"hourlyRate:"<<endl;

            cin>>hourlyRate;

            Worker wr3(id,name,deptId,basicSalary,hoursWorked,hourlyRate);

    int choice;

    do{

    cout<<"Enter your choice: \n 1.Hire Manager\n2. Hire Worker\n3. Display information of all employees net salary\n4. Exit"<<endl;

    cin>>choice;

    switch(choice)

    {

        case 1:

        Employee\* e[3];

            e[0]=&m1;

            e[1]=&m2;

            e[2]=&m3;

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

        for(int i=0;i<3;i++)

        {

            e[i]->display();

        }

        break;

       case 2:

            Employee\* e1[3];

            e1[0]=&wr1;

           e1[1]=&wr2;

           e1[2]=&wr3;

            cout<<"Worker Details: "<<endl;

        for(int i=0;i<3;i++)

        {

            e1[i]->display();

        }

        break;

        case 3:

        Employee\* e2[3];

        e2[0]=&m1;

        e2[1]=&m2;

        e2[2]=&m3;

        cout<<"Manager Salary: "<<endl;

        for(int i=0;i<3;i++)

        {

           cout<<"Net Salary of Manager ("<<i+1<<") : "<<e2[i]->computeNetSalary();

        }

        Employee\* e3[3];

        e3[0]=&wr1;

        e3[1]=&wr2;

        e3[2]=&wr3;

        cout<<"Worker Salary: "<<endl;

        for(int i=0;i<3;i++)

        {

            cout<<"Net Salary of Worker ("<<i+1<<") : "<<e3[i]->computeNetSalary();

        }

        break;

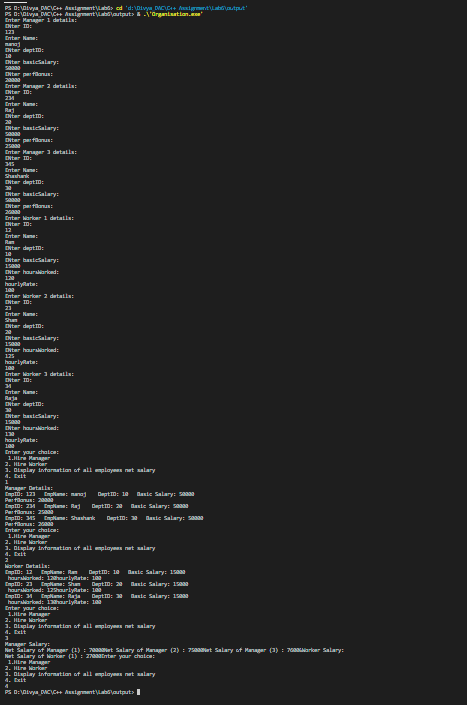
   }

    }while(choice!=4);

return 0;

}

Output:



2:Create cpp application for bank account handling.

2.1. Create a class BankAccount -- acct no(int),customer name(string),balance(double)

Add constr. (2 constrs : first to accept all details )

2.2 Add Business logic methods

Methods

public void withdraw(double amt)

public void deposit(double amt)

2.3: Create object of account class and test withdraw and deposit methods.

Program:

/\*Create cpp application for bank account handling.

2.1. Create a class BankAccount -- acct no(int),customer name(string),balance(double)

Add  constr. (2 constrs : first to accept all details )

2.2 Add Business logic methods

Methods

public void withdraw(double amt)

public void deposit(double amt)

2.3: Create object of account class and test withdraw and deposit methods.\*/

#include<iostream>

using namespace std;

class BankAccount{

    private:

    int accNo;

    string Name;

    double balance;

    public:

    BankAccount()

    {

    accNo=0;;

    Name="\0";

    balance=0;

    }

    BankAccount(int accNo,string Name,double balance)

    {

    this->accNo=accNo;

    this->Name=Name;

    this->balance=balance;

    }

    double getBalance()

    {

        return balance;

    }

    void withdraw(double amt)

    {

       balance=balance-amt;

    }

    void deposit(double amt)

    {

       balance=balance+amt;

    }

    void displayBalance()

    {

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

    }

};

int main()

{

    int accNo;string Name;double balance;

    double amt1;

    cout<<"Enter youe account number"<<endl;

    cin>>accNo;

    cout<<"Enter youe Name"<<endl;

    cin>>Name;

    cout<<"Enter your balance"<<endl;

    cin>>balance;

    BankAccount BA2(accNo,Name,balance);

    int choice;

    do{

        cout<<"Enter your choice: \n1.withdraw money\n2.deposit Money\n3.displayBalance\n4.exit"<<endl;

    cin>>choice;

        switch(choice)

        {

        case 1:

        cout<<"Enter Amount to withdraw"<<endl;

        cin>>amt1;

        if(amt1>BA2.getBalance())

        {

          cout<<"You have insufficient balance "<<BA2.getBalance()<<" please try again later"<<endl;

        }

        else

        {

          BA2.withdraw(amt1);

        }

        break;

        case 2:

        cout<<"Enter Amount to deposit"<<endl;

        cin>>amt1;

        BA2.deposit(amt1);

        break;

        case 3:

        BA2.displayBalance();

        break;

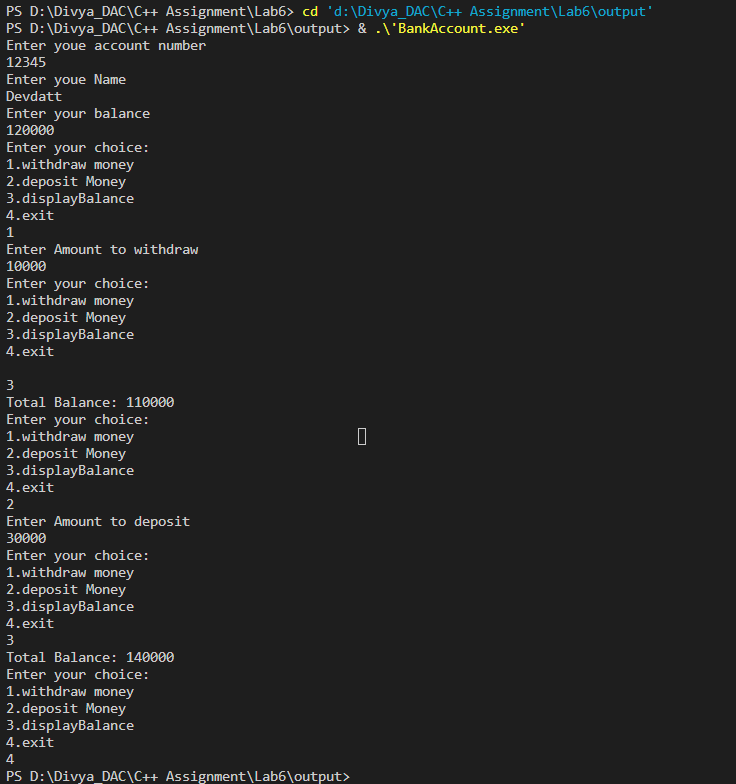
       }

    }while(choice!=4);

    return 0;

}

Output:



3:Create a abstract class Shape with pure virtual method area;

Create Rectangle,Circle,Square class..inherit them from Shape class..Override area method.

Test these all classes by creating object of respective class.

Program:

/\*Create a abstract class Shape with pure virtual method area;

Create Rectangle,Circle,Square class..inherit them from Shape class..Override area method.

Test these all classes by creating object of respective class.\*/

#include<iostream>

using namespace std;

class Shape{

    public:

    virtual void displayArea()=0;

    virtual ~Shape() {}

};

class Rectangle:public Shape{

    private:

    int length;

    int breadth;

    public:

    Rectangle(int length=0,int breadth=0)

    {

        this->length=length;

        this->breadth=breadth;

    }

    void displayArea()

    {

        cout<<" Rectangle Area: "<<(length\*breadth)<<endl;

    }

};

class Circle:public Shape{

    private:

    int radius;

    public:

    Circle(int radius=0)

    {

        this->radius=radius;

    }

    void displayArea()

    {

        cout<<"Circle Area: "<<(3.14\*radius\*radius)<<endl;

    }

};

class Square:public Shape{

    private:

   int side;

    public:

    Square(int side=0)

    {

        this->side=side;

    }

    void displayArea()

    {

        cout<<"Square Area: "<<(side\*side)<<endl;

    }

};

int main()

{

    int choice;

    do{

        cout<<"Enter Your choice: \n1.Area of Rectangle\n2.Area of circle\n3.Area of Square\n4.exit"<<endl;

        cin>>choice;

        switch(choice)

        {

            case 1:

            {

            int length,breadth;

            cout<<"Enter length and breadth: "<<endl;

            cin>>length>>breadth;

            Rectangle R1(length,breadth);

            R1.displayArea();

            break;

            }

            case 2:

            {

            int radius;

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

            cin>>radius;

            Circle c1(radius);

            c1.displayArea();

            break;

            }

            case 3:

            {

            int side;

            cout<<"Enter side of square: "<<endl;

            cin>>side;

            Square s1(side);

            s1.displayArea();

            break;

            }

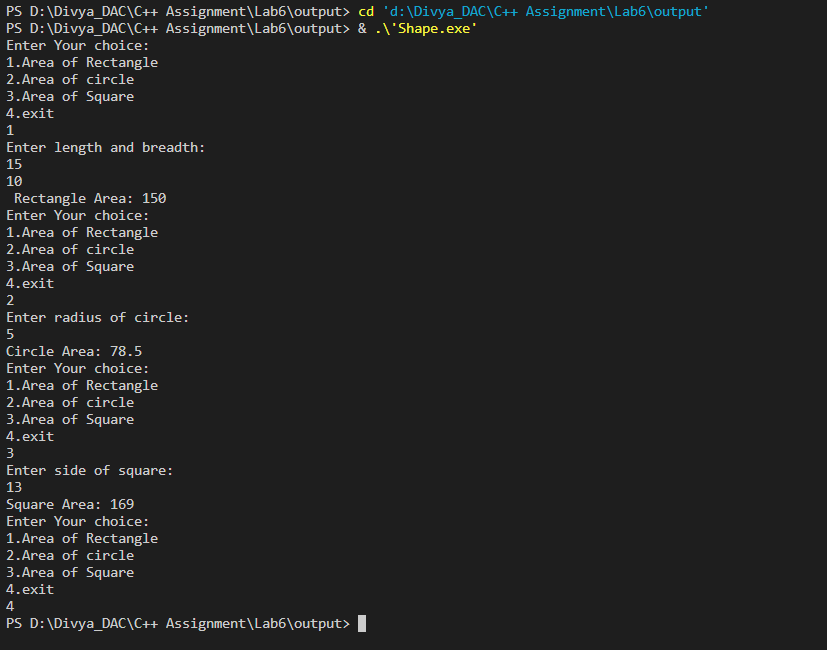
        }

    }while(choice!=4);

    return 0;

}

Output:



**Lab7**

Exception

create Stack class for storing 10 numbers

create function push(int number)--->number will get stored in array

create function pop() will return top most number ,last added number

Note:Hadle StackFull and StackEmpty Exception

Stack s;//array[4];

s.push(10);s.push(20);s.push(30);//s.push(40); s.push(50);

int a=s.pop();//------> 30

a=s.pop();//----->20

a=s.pop();//------>10

Program:

#include<iostream>

using namespace std;

class Exception{

    public:

    string message;

    Exception(string error){

        message = error;

    }

};

class Stack

{

    private:

    int arr[4];

    int top;

    public:

    Stack()

    {

        top = -1;

    }

    void push(int n)

    {

        if(top<3)

        {

        arr[++top]=n;

        }

        else

        {

            throw new Exception("stack is full");

        }

    }

    void print()

    {

        for(int i=0;i<4;i++)

        {

            cout<<arr[i]<<" ";

        }

    }

    int pop()

    {

        if(top != -1)

        {

           return arr[top--];

        }

        else

        {

           throw new Exception("stack is empty");

        }

    }

};

int main()

{

    Stack s;

    int ch;

    int num;

    do{

        cout<<"Enter your choice:\n 1.Push\n2.Pop\n3.display Stack\n4.Exit "<<endl;

        cin>>ch;

        switch(ch)

        {

            case 1:

            try{

            cout<<"Enter Num: "<<endl;

            cin>>num;

            s.push(num);

            }

            catch(Exception \*obj)

           {

           cout<<obj->message<<endl;

           }

        break;

            case 2:

            try{

            cout<<"Popped element: "<<s.pop()<<endl;

            }

            catch(Exception \*obj)

            {

             cout<<obj->message<<endl;

            }

       break;

            case 3:

            s.print();

            break;

        }

    }while(ch!=4);

    return 0;

}

Output:

