**Day 1**

1:write program to test Hello World.

Ans-

#include <iostream>

using namespace std;

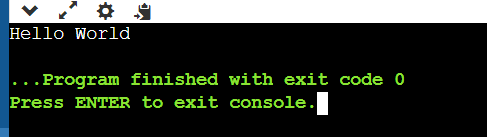
int main() {

// Write C++ code here

cout<<"hello world";

return 0;

}



2:Write a program to adddition of two numbers .

#include <iostream>

using namespace std;

int main()

{

int a=10;

int b=20;

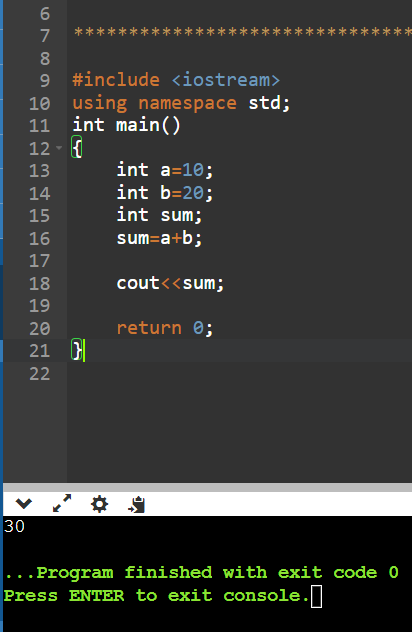
int sum;

sum=a+b;

cout<<sum;

return 0;

}



3:Write a program to swap two numbers.

#include <iostream>

using namespace std;

int main()

{

int a=10;

int b=20;

cout<<" before swapping "<<a<<" "<<b<<endl;

int temp ;

temp =a;

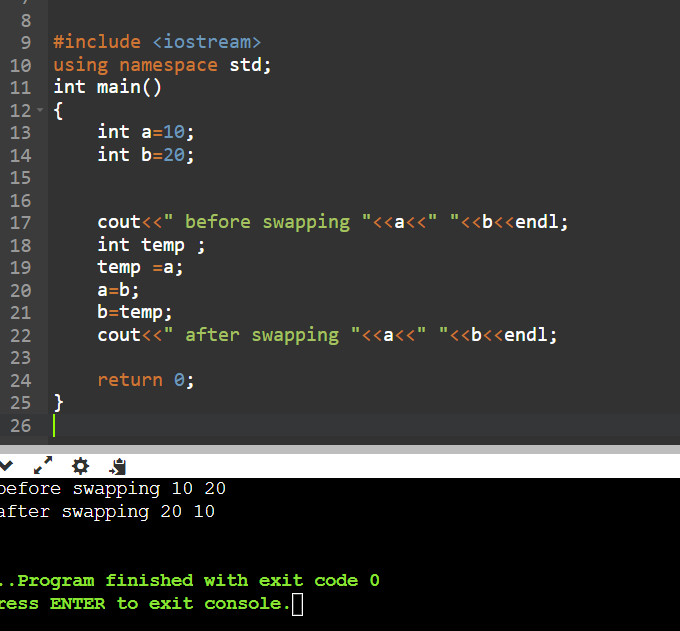
a=b;

b=temp;

cout<<" after swapping "<<a<<" "<<b<<endl;

return 0;

}



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

#include <iostream>

using namespace std;

int main()

{

int num;

cout<<"enter a number "<<endl;

cin>>num;

if(num==0){

cout<<"not even and not odd";

}

if(num%2==0){

cout<<"it is a even number"<<endl;

}

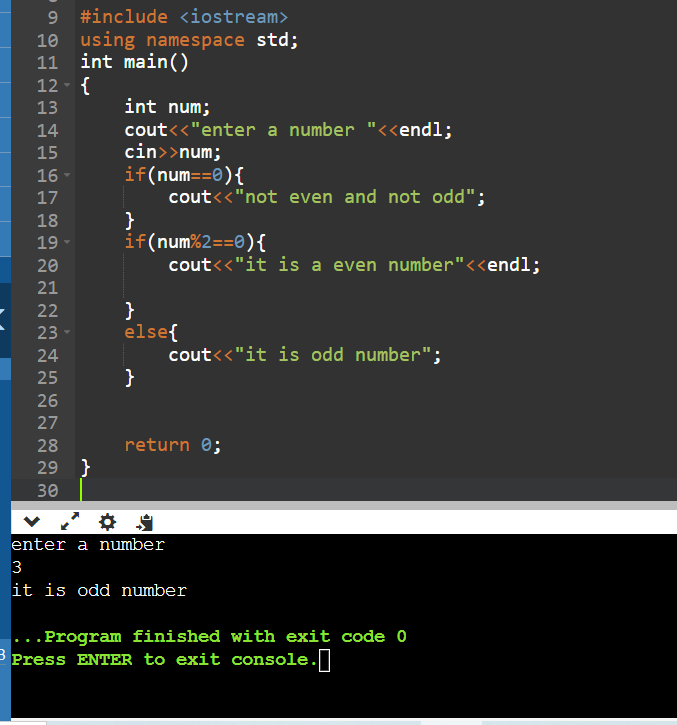
else{

cout<<"it is odd number";

}

return 0;

}



5. 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 num;

cout<<"enter a number "<<endl;

cin>>num;

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

cout<<" num is divisible by 5 and 7";

}

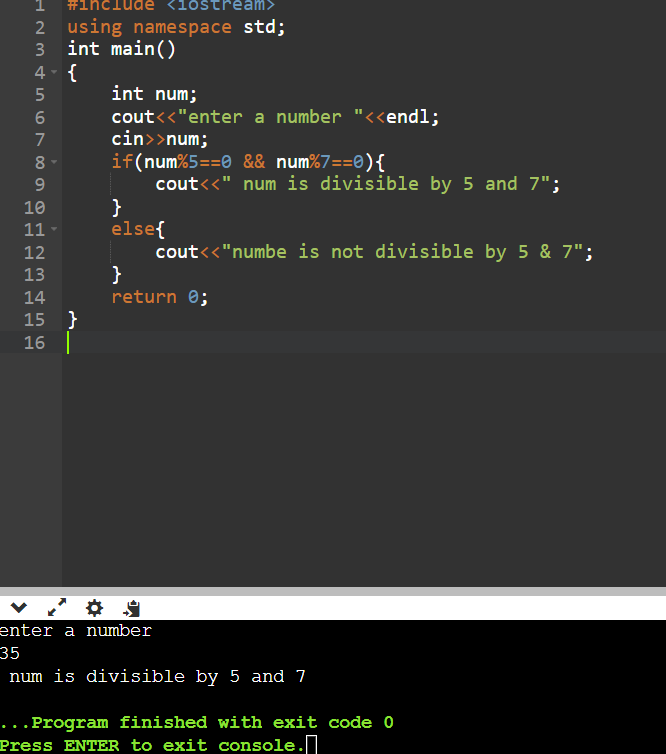
else{

cout<<"numbe is not divisible by 5 & 7";

}

return 0;

}



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%

#include <iostream>

using namespace std;

int main()

{

float salary,tax;

cout<<"enter the emp basic salary :";

cin>>salary;

if(salary<=150000)

{

tax=0;

cout<<"tax ="<<tax;

}

else if(salary>150000 && salary<=300000)

{

tax=(salary\*0.2);

cout<<"tax = "<<tax;

}

else if(salary>300000)

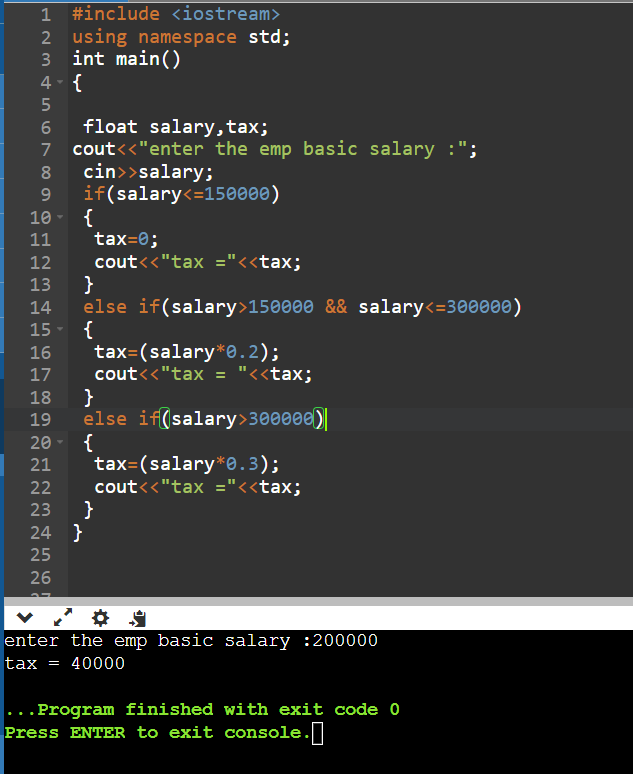
{

tax=(salary\*0.3);

cout<<"tax ="<<tax;

}

}



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)

#include <iostream>

using namespace std;

int main()

{

char c;

int lowercase\_vowel;

cout <<"Enter an alphabet:"<<endl;

cin>>c;

// evaluates to 1 if variable c is a lowercase vowel

lowercase\_vowel = (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');

if (lowercase\_vowel)

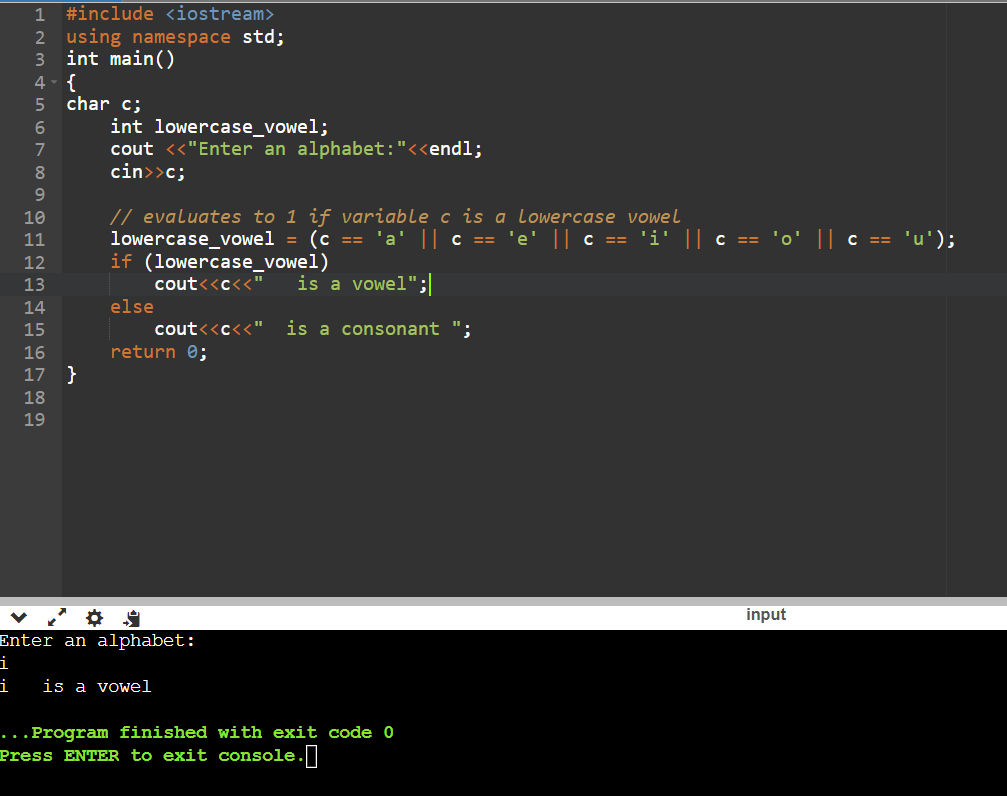
cout<<c<<" is a vowel";

else

cout<<c<<" is a consonant ";

return 0;

}



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

#include <iostream>

using namespace std;

int main()

{

int angle1, angle2, angle3, sum;

cout<<"Enter three angles of triangle:";

cin>>angle1>>angle2 >>angle3;

sum = angle1 + angle2 + angle3;

if(sum == 180 && angle1 > 0 && angle2 > 0 && angle3 > 0)

{

cout<<"Triangle is valid.";

}

else

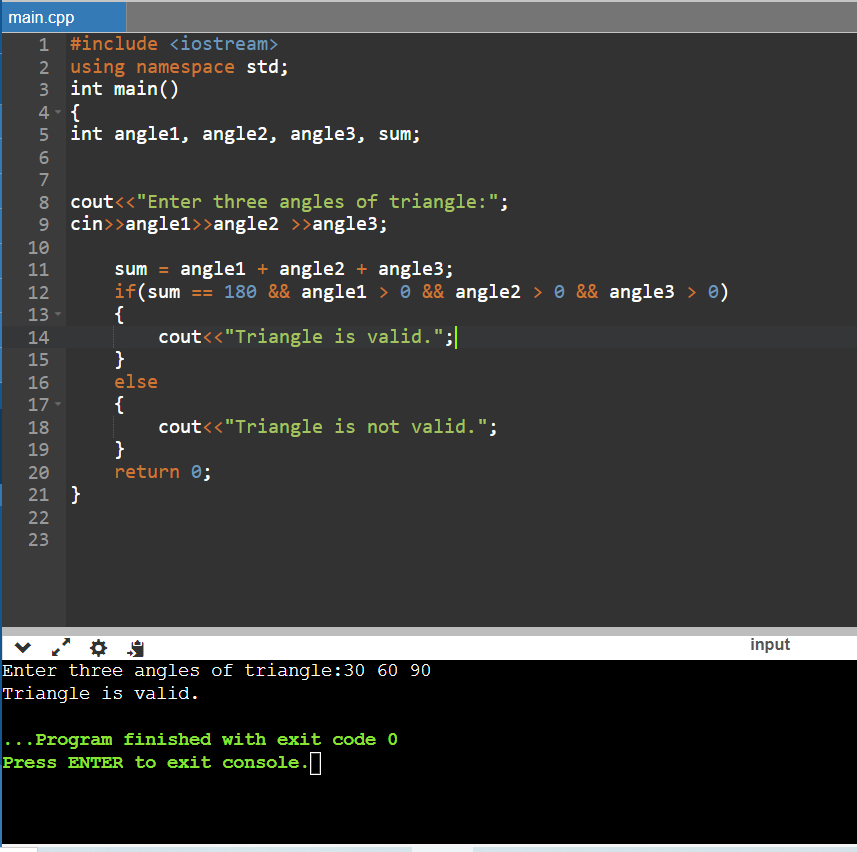
{

cout<<"Triangle is not valid.";

}

return 0;

}



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

#include <iostream>

using namespace std;

int main()

{

int num;

cout<<"enter a number ";

cin>>num;

int fact=1;

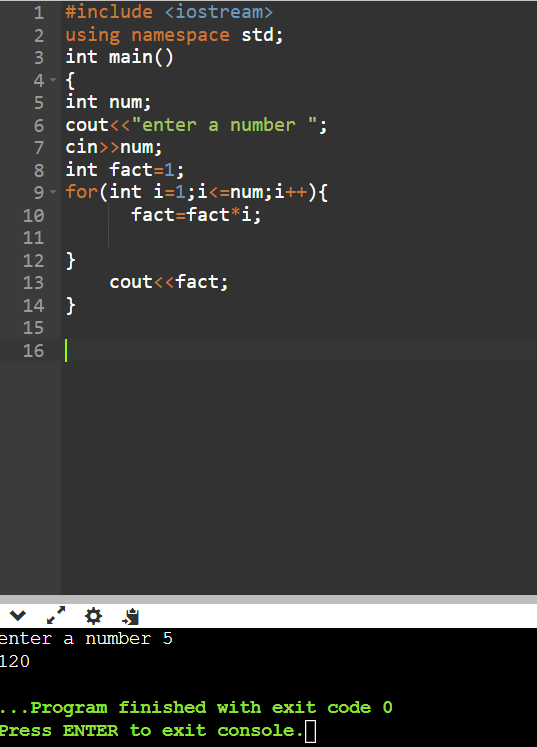
for(int i=1;i<=num;i++){

fact=fact\*i;

}

cout<<fact;

}



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

#include <iostream>

using namespace std;

int main()

{

int exponent;

float base, result = 1;

cout << "Enter base and exponent respectively: ";

cin >> base >> exponent;

cout << base << "^" << exponent << " = ";

while (exponent != 0) {

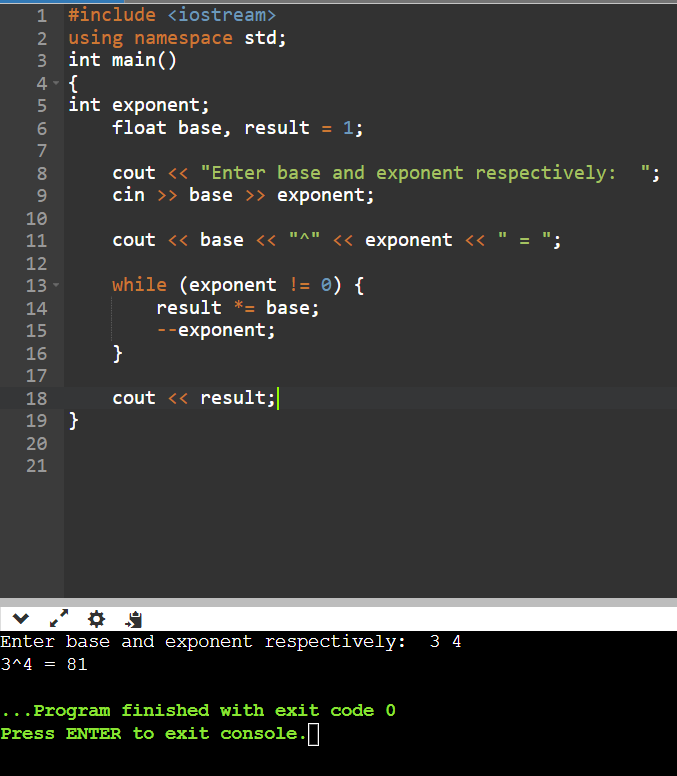
result \*= base;

--exponent;

}

cout << result;

}



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

#include <iostream>

using namespace std;

int main()

{

int n, i, flag = 0;

cout<<"Enter a positive integer: ";

cin>>n;

if (n == 0 || n == 1)

flag = 1;

for (i = 2; i <= n / 2; ++i) {

if (n % i == 0) {

flag = 1;

break;

}

}

if (flag == 0)

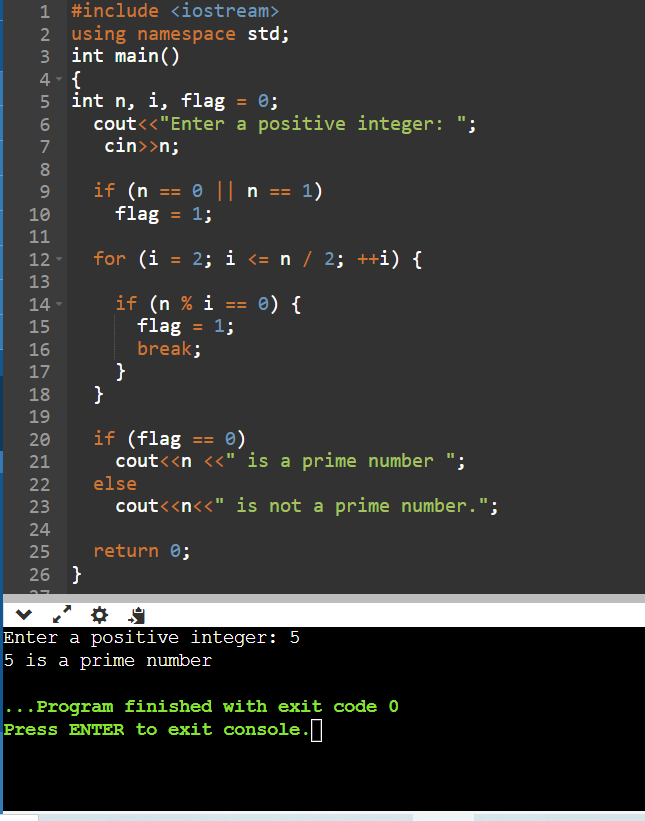
cout<<n <<" is a prime number ";

else

cout<<n<<" is not a prime number.";

return 0;

}



12:Sum of series :

1+2+3+….+n

#include <iostream>

using namespace std;

int main()

{

int i,n,sum=0;

cout<<"1+2+3+……+n"<<endl;

cout<<"Enter the value of n:";

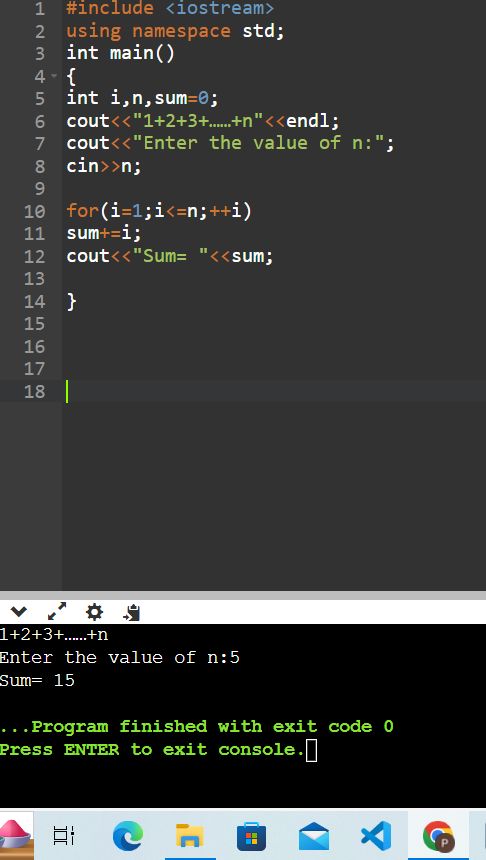
cin>>n;

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

sum+=i;

cout<<"Sum= "<<sum;

}



13:Check whether the number is palindrome or not?

#include <iostream>

using namespace std;

int main()

{

int n, num, digit, rev = 0;

cout << "Enter a positive number: ";

cin >> num;

n = num;

do

{

digit = num % 10;

rev = (rev \* 10) + digit;

num = num / 10;

} while (num != 0);

cout << " The reverse of the number is: " << rev << endl;

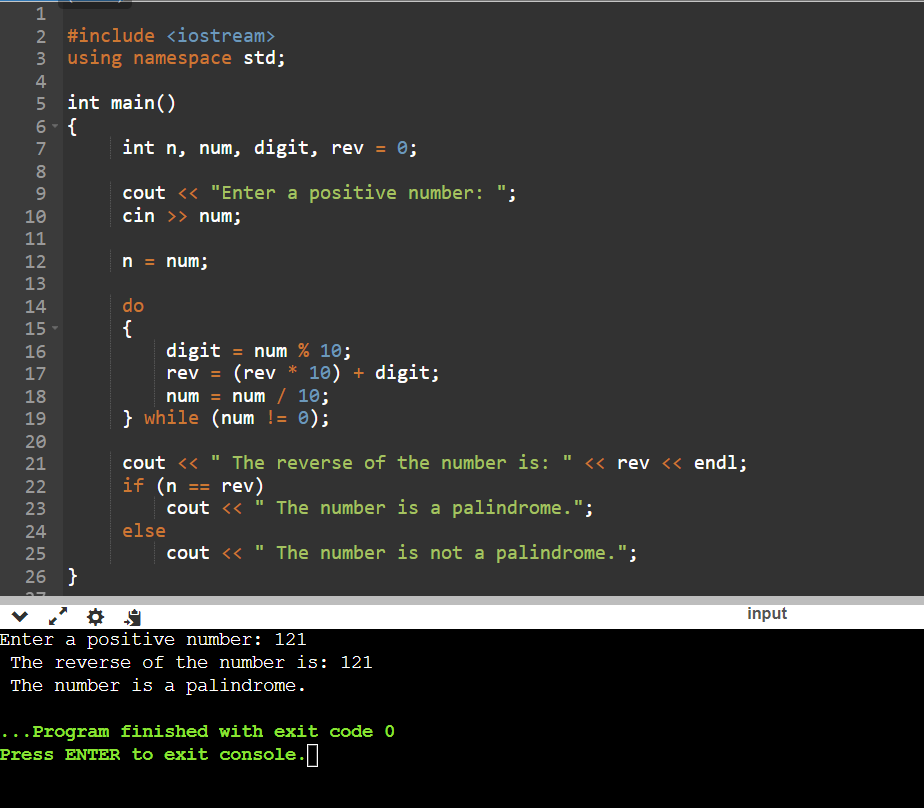
if (n == rev)

cout << " The number is a palindrome.";

else

cout << " The number is not a palindrome.";

}



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 i, number, Even\_Sum = 0, Odd\_Sum = 0;

cout<<"\nPlease Enter the Maximum Limit Value : ";

cin>>number;

for(i = 1; i <= number; i++)

{

if ( i%2 == 0 )

{

Even\_Sum = Even\_Sum + i;

}

else

{

Odd\_Sum = Odd\_Sum + i;

}

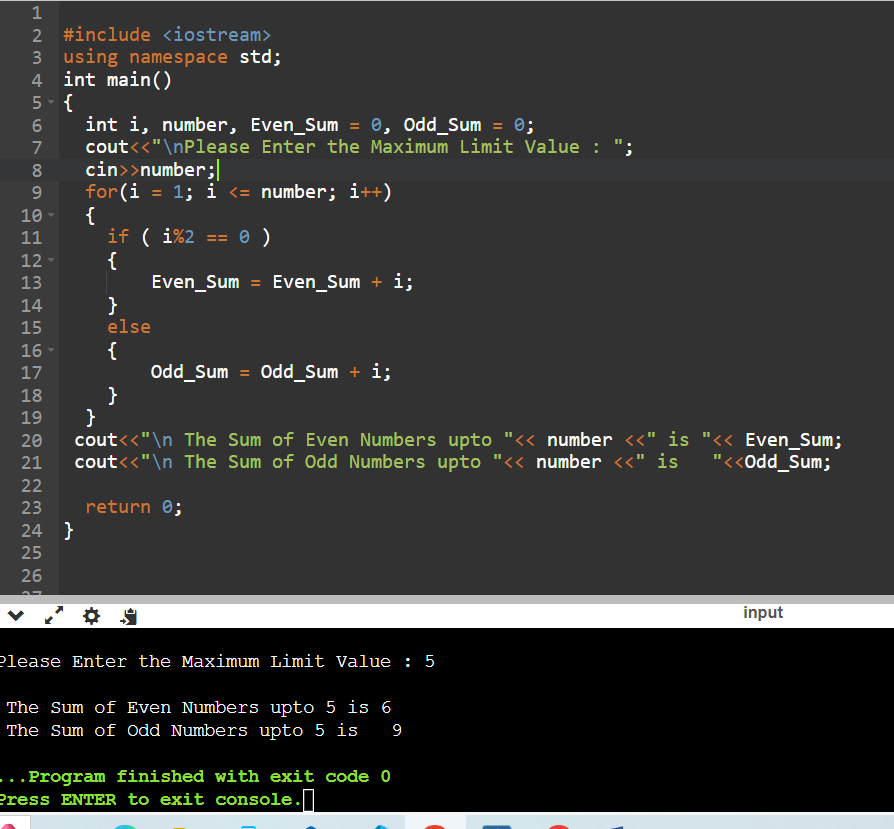
}

cout<<"\n The Sum of Even Numbers upto "<< number <<" is "<< Even\_Sum;

cout<<"\n The Sum of Odd Numbers upto "<< number <<" is "<<Odd\_Sum;

return 0;

}



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

#include <iostream>

using namespace std;

int main() {

int n, reversed\_number = 0, remainder;

cout << "Enter an integer: ";

cin >> n;

while(n != 0) {

remainder = n % 10;

reversed\_number = reversed\_number \* 10 + remainder;

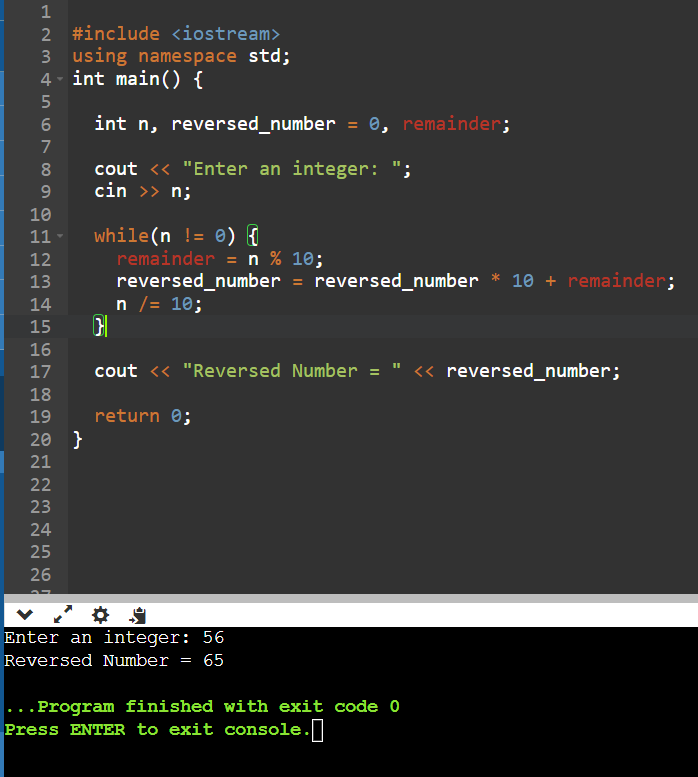
n /= 10;

}

cout << "Reversed Number = " << reversed\_number;

return 0;

}



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

#include <iostream>

using namespace std;

bool isPrime(int n)

{

if(n == 1 || n == 0) return false;

for(int i = 2; i < n; i++)

{

if(n % i == 0) return false;

}

return true;

}

int main()

{

int N;

cout<<"enter a number ";

cin>>N;

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

{

if(isPrime(i))

{

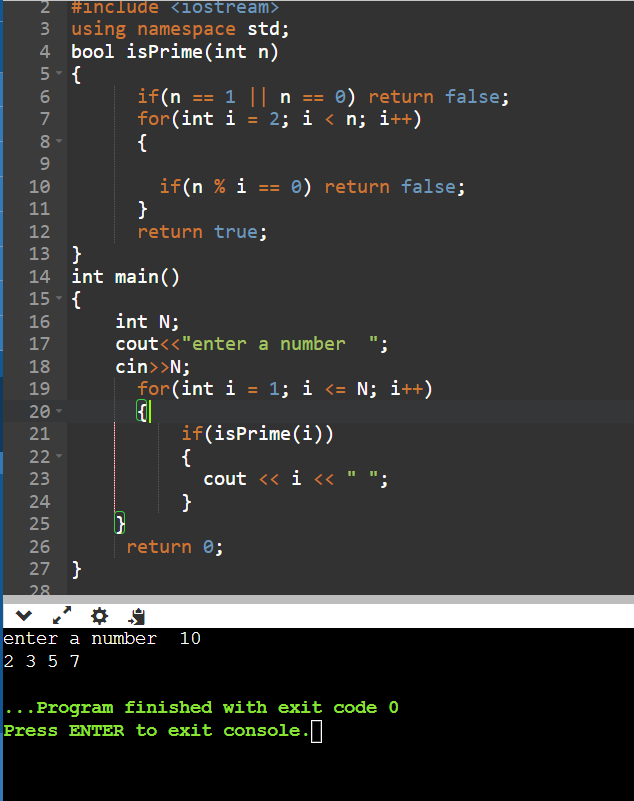
cout << i << " ";

}

}

return 0;

}



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

#include <iostream>

using namespace std;

int main() {

int num, originalNum, remainder, result = 0;

cout << "Enter a three-digit integer: ";

cin >> num;

originalNum = num;

while (originalNum != 0) {

remainder = originalNum % 10;

result += remainder \* remainder \* remainder;

originalNum /= 10;

}

if (result == num)

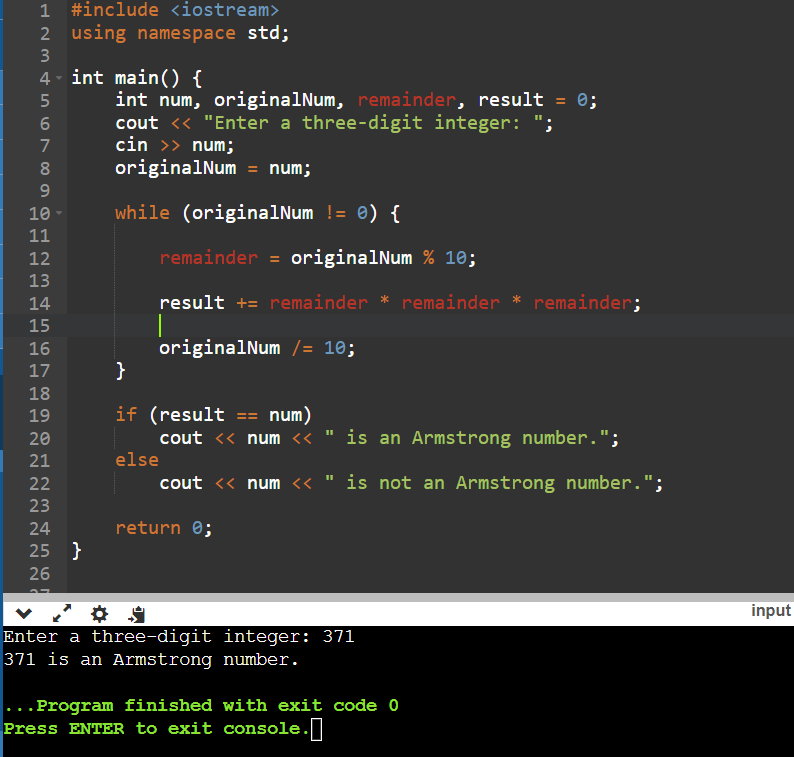
cout << num << " is an Armstrong number.";

else

cout << num << " is not an Armstrong number.";

return 0;

}



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

#include <iostream>

using namespace std;

int main() {

double n1, n2, n3;

cout << "Enter three numbers: ";

cin >> n1 >> n2 >> n3;

if(n1 >= n2 && n1 >= n3)

cout << "Largest number: " << n1;

else if(n2 >= n1 && n2 >= n3)

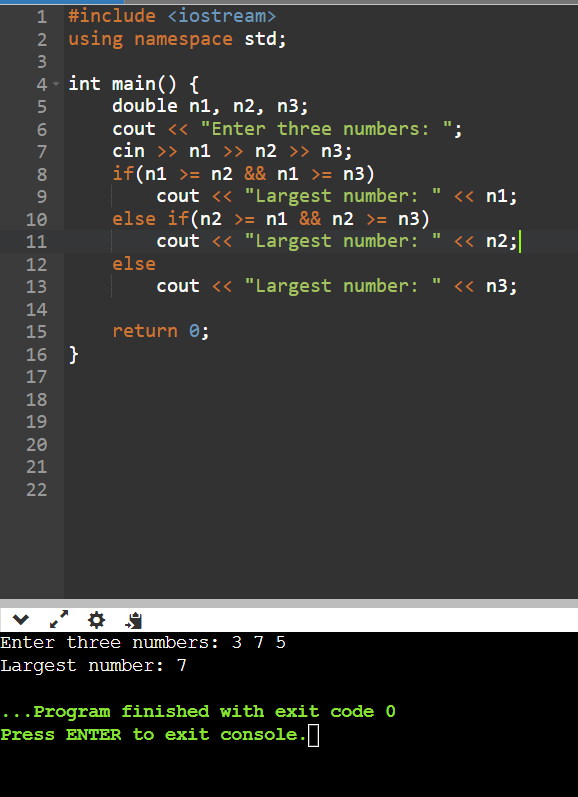
cout << "Largest number: " << n2;

else

cout << "Largest number: " << n3;

return 0;

}



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

20: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()

{

int n, num = 0;

cout << "\n\n Print a number in words:\n";

cout << "-----------------------------\n";

cout << " Input any number: ";

cin >> n;

switch (n) {

case 0:

cout << "Zero ";

break;

case 1:

cout << "One ";

break;

case 2:

cout << "Two ";

break;

case 3:

cout << "Three ";

break;

case 4:

cout << "Four ";

break;

case 5:

cout << "Five ";

break;

case 6:

cout << "Six ";

break;

case 7:

cout << "Seven ";

break;

case 8:

cout << "Eight ";

break;

case 9:

cout << "Nine ";

break;

}

cout << endl;

}

21. 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()

{

int a,b,res;

char c;

cout<<"Enter any one operator +, -, \*, \n";

cin>>c;

cout<<"\n Enter two numbers \n";

cin>>a>>b;

switch(c)

{

case '+': res=a+b;

cout<<" The sum is "<<res;

break;

case '-': res=a-b;

cout<<"The difference is"<<res;

break;

case '\*': res=a\*b;

cout<<" The product is "<<res;

break;

case '/': res=a/b;

cout<<" The quotient is "<<res;

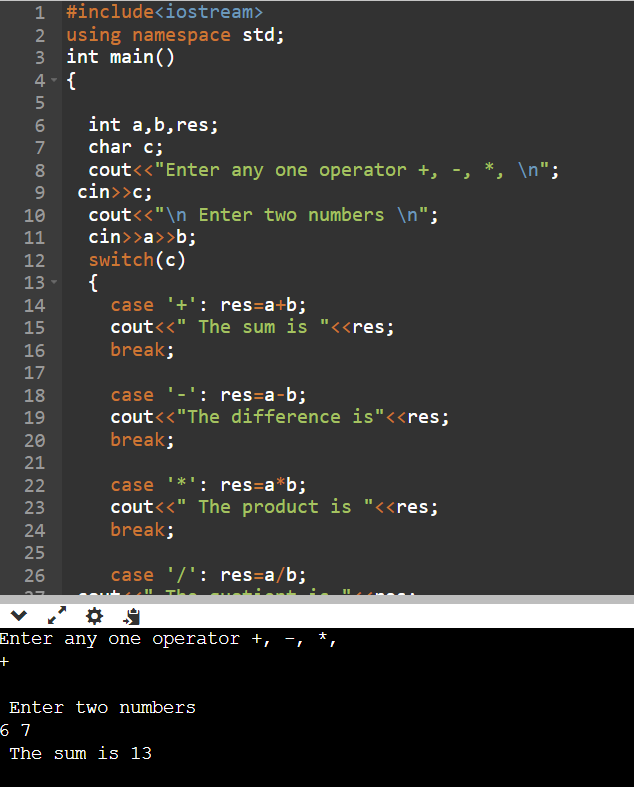
break;

default: cout<<" Invalid entry";

}

return 0;

}



Day 2

Part 1

1: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 n, sum = 0;

cout << "Enter a positive integer: ";

cin >> n;

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

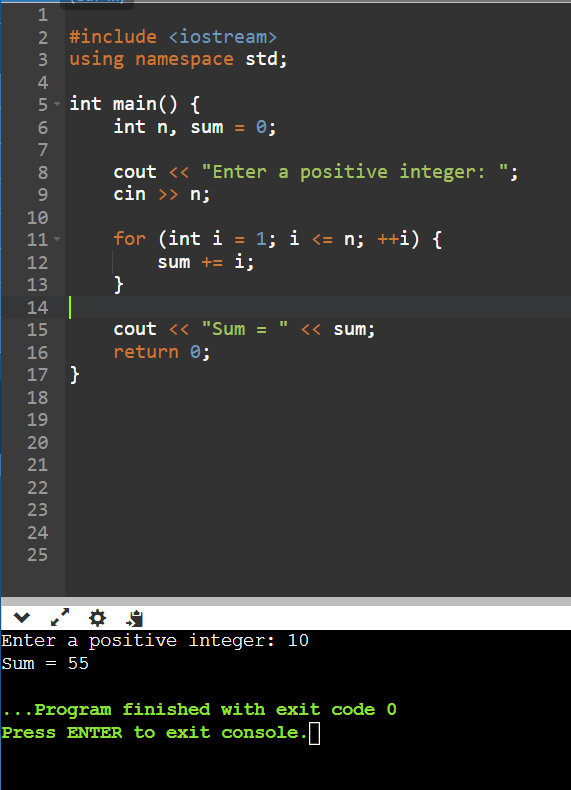
sum += i;

}

cout << "Sum = " << sum;

return 0;

}



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

#include<iostream>

using namespace std;

int main()

{

long int x,n,pow=1,j=1;

cout<<"Enter the values of X and n : ";

cin>>x>>n;

pow=x;

while(n!=j)

{

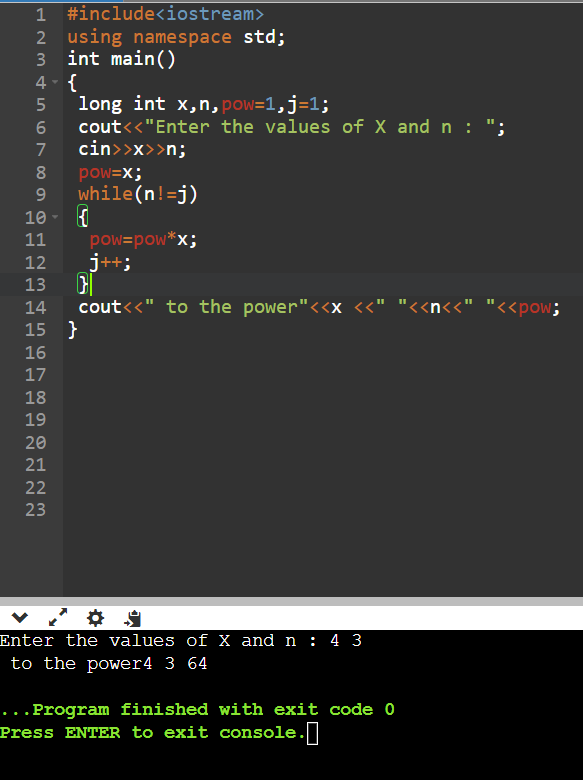
pow=pow\*x;

j++;

}

cout<<" to the power"<<x <<" "<<n<<" "<<pow;

}



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

#include<iostream>

using namespace std;

int display(char);

int main()

{

char ch,c;

cout<<"Enter character:";

cin>>ch;

display(ch);

}

int display(char ch)

{

int n,i;

cout<<"how many next char:";

cin>>n;

cout<<"\n You entered: \n "<<ch;

cout<<" Next character : ";

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

{

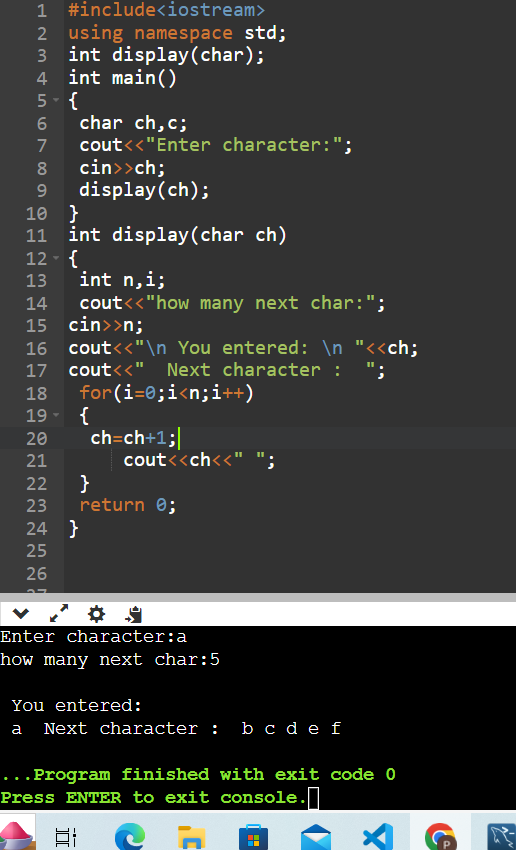
ch=ch+1;

cout<<ch<<" ";

}

return 0;

}



4. 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() {

int n;

long factorial = 1.0;

cout << "Enter a positive integer: ";

cin >> n;

if (n < 0)

cout << "Error! Factorial of a negative number doesn't exist.";

else {

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

factorial \*= i;

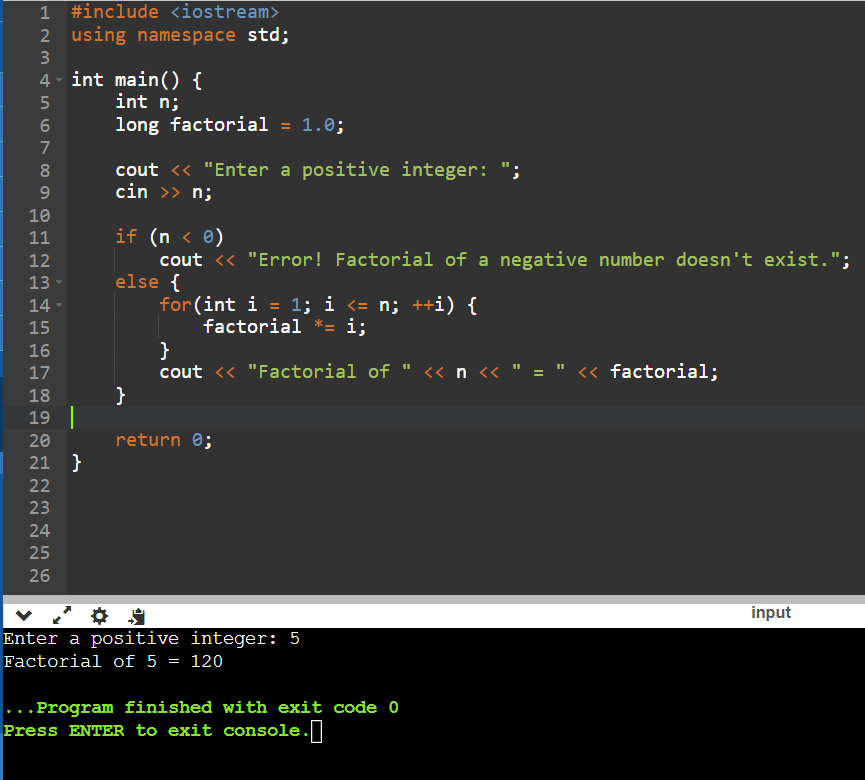
}

cout << "Factorial of " << n << " = " << factorial;

}

return 0;

}



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

#include <iostream>

using namespace std;

int main() {

int num, i;

cout<<"Enter a positive integer: ";

cin>>num;

cout<<"Factors of are: ";

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

if (num % i == 0) {

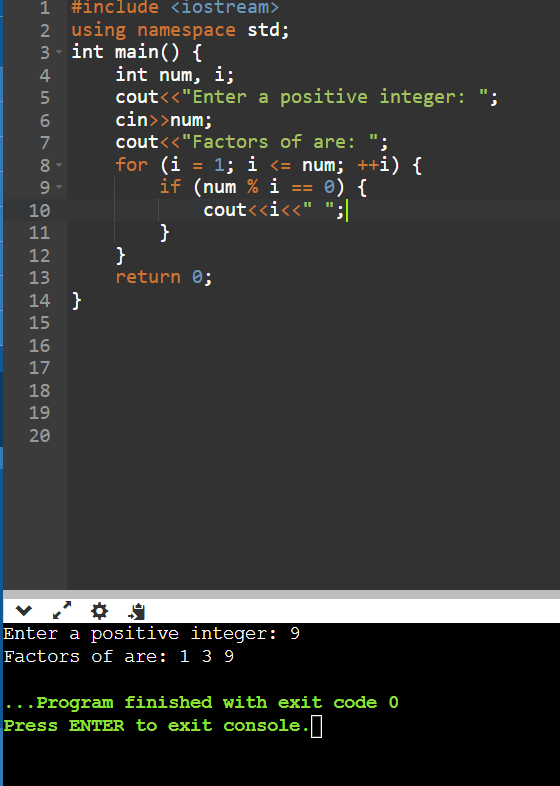
cout<<i<<" ";

}

}

return 0;

}



6. Accept two numbers and calculate GCD of them.

#include <bits/stdc++.h>

using namespace std;

int gcd(int a, int b)

{

int result = min(a, b);

while (result > 0) {

if (a % result == 0 && b % result == 0) {

break;

}

result--;

}

return result;

}

int main()

{

int a,b;

cout<<"enter two numbers";

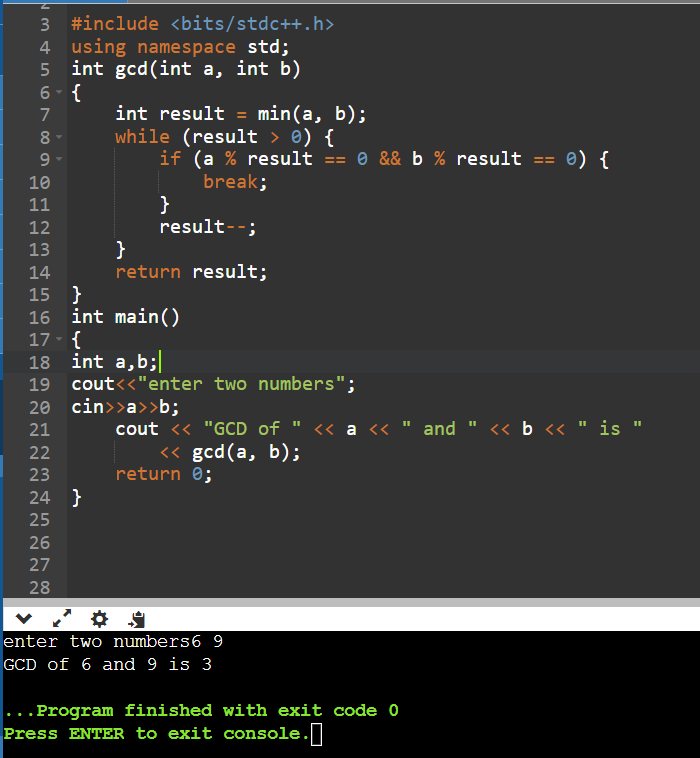
cin>>a>>b;

cout << "GCD of " << a << " and " << b << " is "

<< gcd(a, b);

return 0;

}



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.

#include<iostream>

using namespace std;

int main ()

{

int choice,r,l,w,b,h;

float area;

cout<<"Input 1 for area of circle";

cout<<"Input 2 for area of rectangle\n";

cout<<"Input 3 for area of triangle\n";

do{

cout<<"\nInput your choice : ";

cin>>choice;

switch(choice)

{

case 1:

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

cin>>r;

area=3.14\*r\*r;

cout<<"The area is : "<<area;

break;

case 2:

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

cin>>l>>w;

area=l\*w;

cout<<"The area is : "<<area;

break;

case 3:

cout<<"Input the base and height of the triangle : ";

cin>>b>>h;

area=.5\*b\*h;

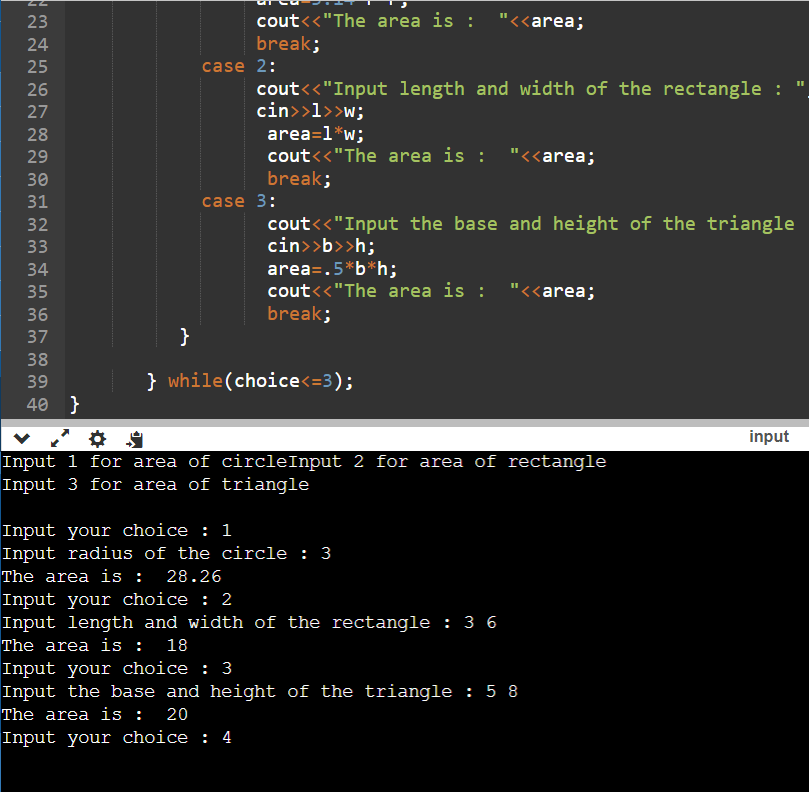
cout<<"The area is : "<<area;

break;

}

} while(choice<=3);

}



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

Day 2

Part2

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.

#include <iostream>

using namespace std;

int main()

{

int a[8],i,s=0,g,l;

float avg;

cout<<"Enter 8 Numbers:\n";

for(i=0;i<8;i++)

{

cin>>a[i];

s=s+a[i];

avg=s/8.0;

}

cout<<"Sum of Array Elements = "<<s<<endl;

cout<<"Average of Elements = "<<avg<<endl;

g=a[0];

for(i=0;i<8;i++)

if(a[i]>g)

g=a[i];

cout<<"Greatest Element = "<<g<<endl;

l=a[0];

for(i=0;i<8;i++)

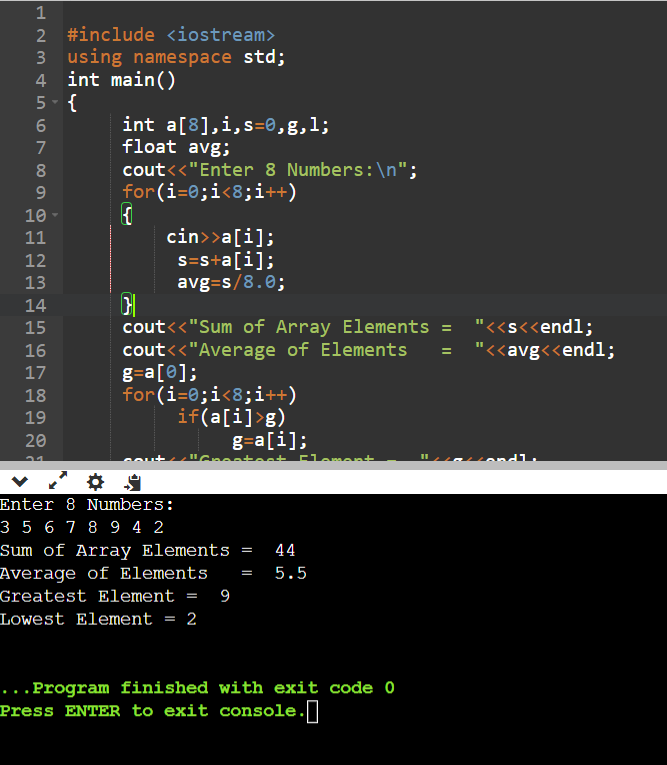
if(a[i]<l)

l=a[i];

cout<<"Lowest Element = "<<l<<endl;

return 0;

}



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

#include <iostream>

using namespace std;

int main()

{

int n,sum=0,m;

cout<<"Enter a number: ";

cin>>n;

while(n>0)

{

m=n%10;

sum=sum+m;

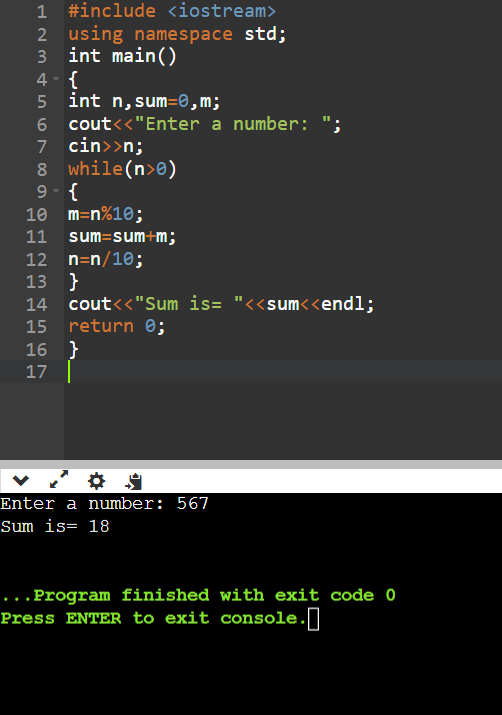
n=n/10;

}

cout<<"Sum is= "<<sum<<endl;

return 0;

}



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

#include<iostream>

using namespace std;

int main()

{

int arr[10], i, eve=0, odd=0;

cout<<"Enter any 10 numbers: ";

for(i=0; i<10; i++)

cin>>arr[i];

for(i=0; i<10; i++)

{

if(arr[i]%2==0)

eve = eve+arr[i];

else

odd = odd+arr[i];

}

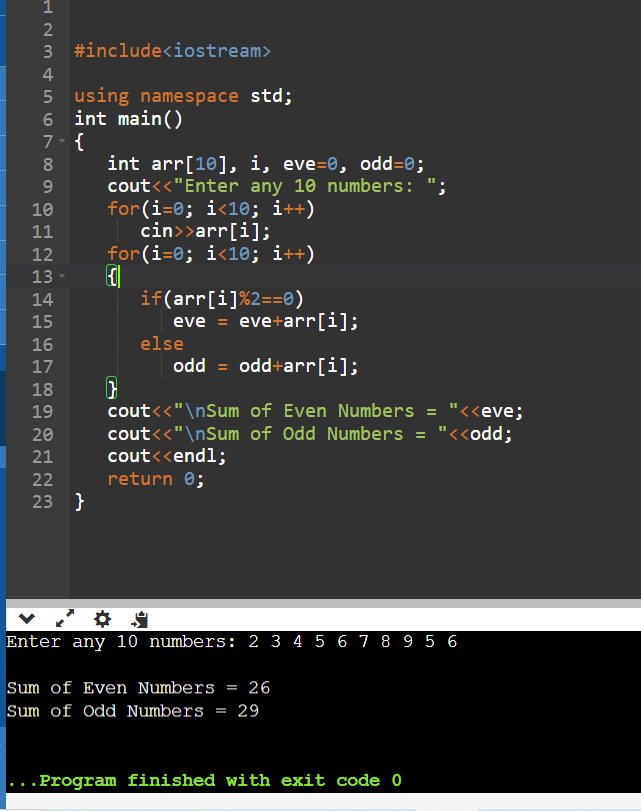
cout<<"\nSum of Even Numbers = "<<eve;

cout<<"\nSum of Odd Numbers = "<<odd;

cout<<endl;

return 0;

}



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

#include <bits/stdc++.h>

using namespace std;

bool isPrime(int n)

{

if(n == 1 || n == 0) return false;

for(int i = 2; i < n; i++)

{

if(n % i == 0) return false;

}

return true;

}

int main()

{

int N;

cout<<"enter a num";

cin>>N;

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

{

if(isPrime(i))

{

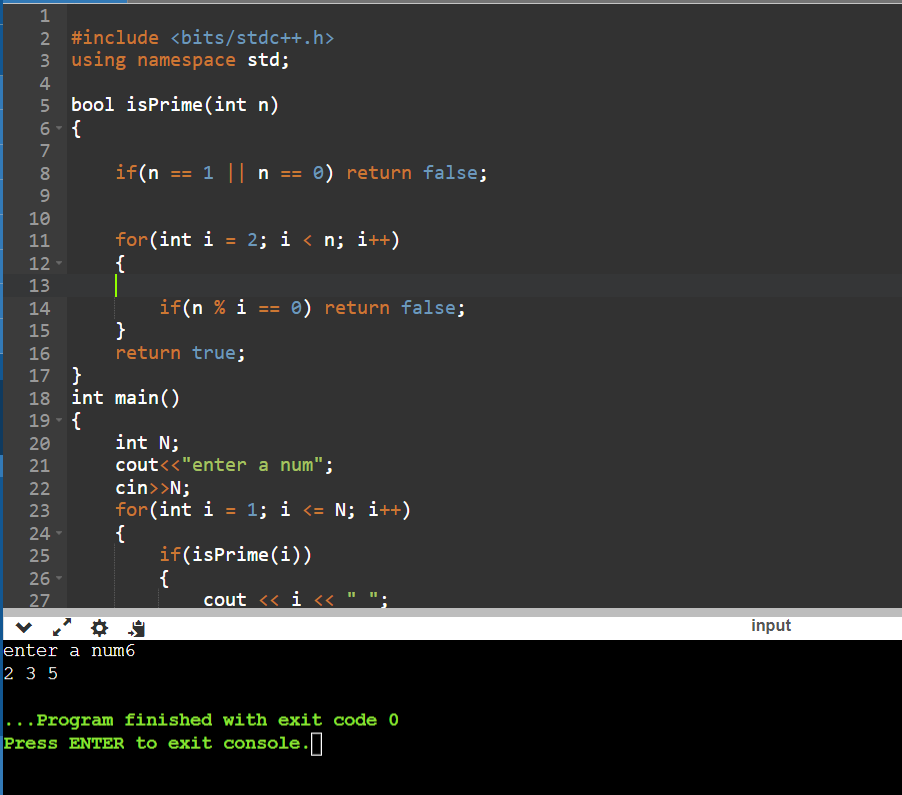
cout << i << " ";

}

}

return 0;

}



5: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;

int main()

{

int a[100],n,i,x;

cout<<"\nEnter The Limit: ";

cin>>n;

cout<<"\nEnter The Value : ";

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

{

cin>>a[i];

}

cout<<"\nEnter The Value to Search :";

cin>>x;

for(i=0;i<n;i++)//1 2 3 4 5

{

if(a[i]==x)

{

cout<<"Value Found @"<<i;

return 0;

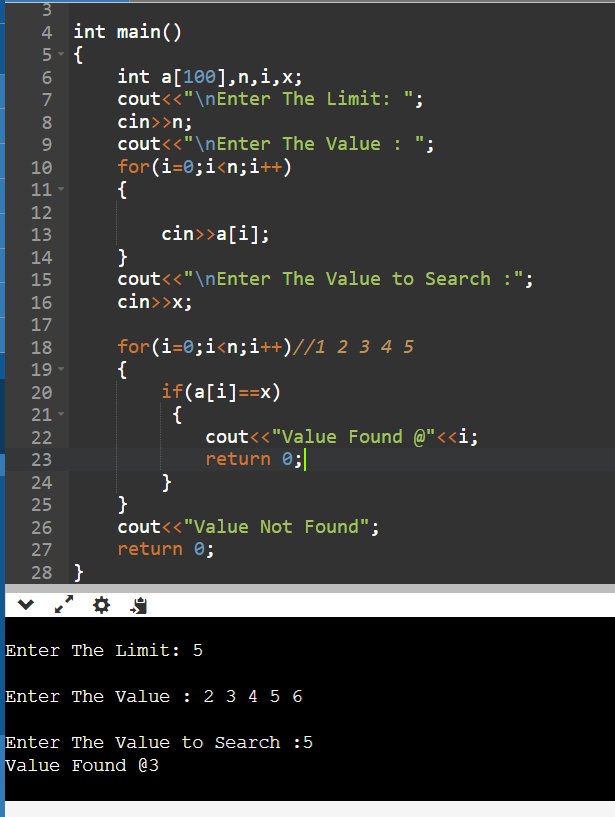
}

}

cout<<"Value Not Found";

return 0;

}



6:Write a program to print following pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

#include <iostream>

using namespace std;

void print(int n)

{

if(n == 0)

{

return;

}

cout << "\* ";

print(n - 1);

}

void pattern(int n)

{

if (n == 0)

{

return;

}

else

{

pattern(n - 1);

print(n);

cout << endl;

}

}

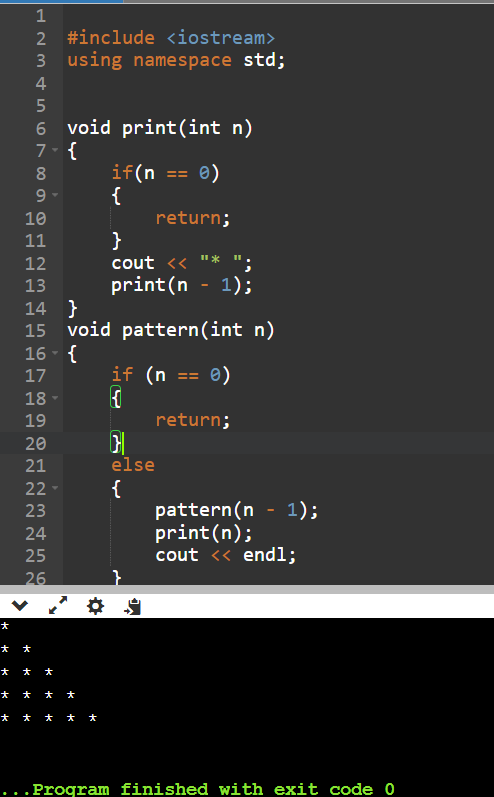
int main()

{

int n = 5;

pattern(n);

}



Day 3

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.

Ans-

#include<iostream>

using namespace std;

class student {

    private :

    int rollno;

    int mark1;

    int mark2;

    int mark3;

    public:

void acceptinfo(){

         cout << "Enter Roll Number: ";

        cin >> rollno;

        cout << "Enter Marks in Subject 1: ";

        cin >> mark1;

        cout << "Enter Marks in Subject 2: ";

        cin >> mark2;

        cout << "Enter Marks in Subject 3: ";

        cin >> mark3;

}

    void display(){

        cout<<"\nrollno:  "<<rollno<<"\nmark1:  "<<mark1<<"\nmark2:  "<<mark2<<"\nmark3:  "<<mark3;

    }

int calTotal(){

  return mark1+mark2+mark3;

}

float calpercentage(){

    float total=calTotal();

    return (total/3.0);

}

char calculateGrade() {

        float percentage=calpercentage();

        if (percentage >= 90)

            return 'A';

        else if (percentage >= 80)

            return 'B';

        else if (percentage >= 70)

            return 'C';

        else if (percentage >= 60)

            return 'D';

        else

            return 'F';

    }

};

int main(){

student s;

s.acceptinfo();

s.display();

int totalmarks=s.calTotal();

float percentage=s.calpercentage();

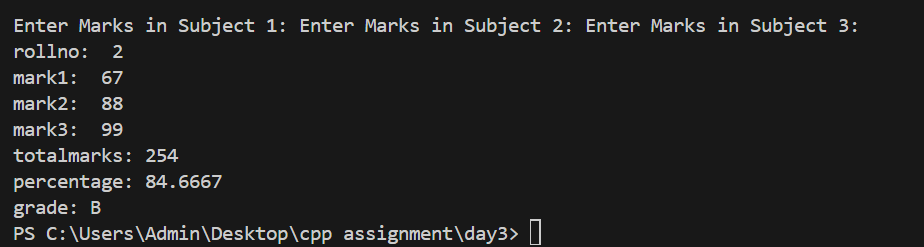
char grade=s.calculateGrade();

cout<<"\ntotalmarks: "<<totalmarks;

cout<<"\npercentage: "<<percentage;

cout<<"\ngrade: "<<grade;

}



1. 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.

#include<iostream>

using namespace std;

class person{

    private:

    string name;

    int age;

    string city;

    public:

    person(){

        name="pratibha";

        age=24;

        city="pune";

    }

    person(string name,int age,string city){

        this->name=name;

        this->age=age;

        this->city=city;

    }

void setage(int a){

    age=a;

}

int getAge(){

return age;

}

void setName(string n){

    name=n;

}

string getame(){

    return name;

}

void display(){

    cout<<" name: "<<name<<"  age: "<<age<<"  city: "<<city;

}

void setCity(string c)

{

    city=c;

}

string getcity(){

    cout<<city;

}

};

int main(){

person p("shrui",45,"pune   ");

p.display();

p.setage(25);

cout<<p.getAge();

p.setName(" tejas ");

cout<<p.getame();

p.setCity(" mumbai ");

p.getcity();

person p1;

p1.display();

}



2. 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 day,month,year;

    public:

    date()

    {

        day=1;month=1;year=2000;

    }

    date(int d,int m,int y)   //parameterize constrctor

    {

        this->day=d;

        this->month=m;

        this->year=y;

    }

    void accept\_date(int d,int m,int y){

    this->day=d;

    this->month=m;

    this->year=y;

    }

    void setd(int d)

    {

        this->day=d;

    }

    void setm(int m)

    {

        this->month=m;

    }

    void sety(int y)

    {

        this->year=y;

    }

    int getd()

    {

        return day;

    }

    int getm()

    {

        return month;

    }

    int gety()

    {

        return year;

    }

    void show\_date()

    {

        cout<<"date:"<<day<<"/"<<month<<"/"<<year<<endl;

    }

};

int main()

{

    date d;

    d.accept\_date(2,3,1999);

    d.show\_date();

    cout<<"Using setter and getter"<<endl;

    d.setd(3);

    d.setm(3);

    d.sety(2000);

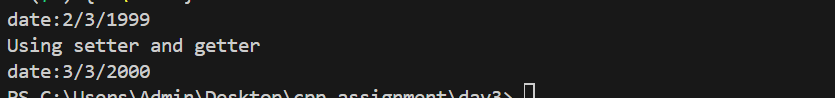
    d.getd();

    d.getm();

    d.gety();

    d.show\_date();

    }



3. 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 bid;

    string auther;

    int price;

    public:

    book()

    {

        bname="concepts of cpp";

        bid=9;

        auther="denis ritche";

        price=200;

    }

  book(string name1,int id, string auth,int p){

    bname=name1;

    bid=id;

    auther=auth;

    price=p;

  }

  void setName(string n){

    bname=n;

}

string getname(){

    return bname ;

}

void setId(int id){

    bid=id;

}

int getId(){

    return bid ;

}

void setAuther(string au){

    auther=au;

}

string getAuther(){

    return auther ;

}

void setPrice(int pr){

    price=pr;

}

int getPrice(){

    return price ;

}

void display(){

    cout<<" bookname: "<<bname<<" bookid : "<<bid<<" auther "<<auther<<" price : "<<price<<endl;

}

}

;

int main(){

book b;

b.display();

b.setName("oops");

cout<<"  setterbname: "<<b.getname()<<endl;

b.setId(3);

cout<<" setterid: "<<b.getId()<<endl;

b.setAuther("rs agrawal");

cout<<" setterauth: "<<b.getAuther()<<endl;

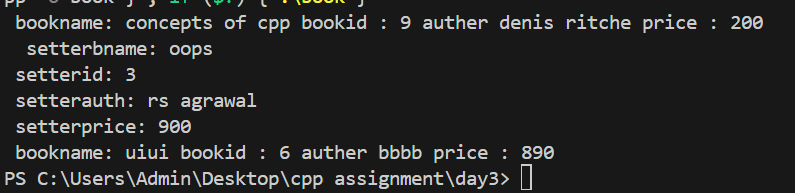
b.setPrice(900);

cout<<" setterprice: "<<b.getPrice()<<endl;

book b1("uiui",6,"bbbb",890);

b1.display();

}



4. 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()

    {

        x=4;

        y=9;

    }

  point(int a,int b){

    x=a;

    y=b;

  }

  void setx(int n){

    x=n;

}

int getx(){

    return x ;

}

void sety(int o){

    y=o;

}

int gety(){

    return y ;

}

void display(){

    cout<<" x :"<<x<<" y: "<<y<<endl;

}

};

int main(){

point p;

p.display();

p.setx(8);

cout<<" x :"<<p.getx()<<endl;

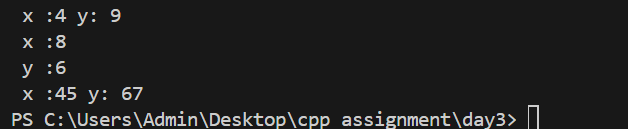
p.sety(6);

cout<<" y :"<<p.gety()<<endl;

point p1(45,67);

p1.display();

}



5. 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 complex{

    private:

    int real;

    int imaginary;

    public:

    complex()

    {

        real=4;

        imaginary=9;

    }

  complex(int a,int b){

    real=a;

    imaginary=b;

  }

  void setreal(int n){

    real=n;

}

int getreal(){

    return real ;

}

void setimaginary(int o){

    imaginary=o;

}

int getimaginary(){

    return imaginary ;

}

void display(){

    cout<<" real :"<<real<<" imaginary: "<<imaginary<<endl;

}

};

int main(){

complex c;

c.display();

c.setreal(88);

cout<<" real :"<<c.getreal()<<endl;

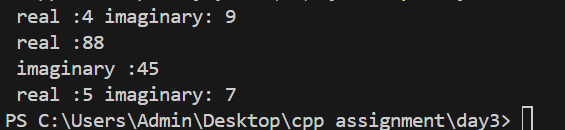
c.setimaginary(45);

cout<<" imaginary :"<<c.getimaginary()<<endl;

complex c1(5,7);

c1.display();

}



Day 4

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{

   protected:

   string movement=" can move ";

   string breath=" can breath ";

   string eat=" can eat ";

   public:

 void Display(){

    cout<<movement<<breath<<eat;

 }

};

class mammel:public Animal{

    private:

   int legs=2;

   string walk=" can walk";

   public: void Display1(){

   cout<<"legs:"<<legs<<walk;

   }};

   class Bird:public Animal{

    private:

    int legs=2;

   string fly=" can fly";

  public: void Display2(){

    cout<<"legs:"<<legs<<fly;

   }};

class Fish:public Animal{

    private:

   int gills=2;

   string swim="can swim";

  public: void Display3(){

    cout<<gills<<swim;

   }

};

int main(){

//Animal a;

//a.Display();

mammel m;

m.Display();

m.Display1();

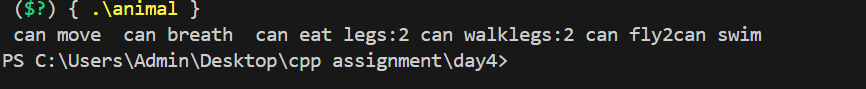
Bird b;

b.Display2();

Fish f;

f.Display3();

}



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{

  protected: int area,perimeter;

};

class rectangle :public shape

{

    private:int length=9,breadth=10;

    public:

    void area(){

        int area=length\*breadth;

        cout<<"area="<<area;

    }

   void perimeter(){

    int per=2\*(length+breadth);

    cout<<"\nperimeter="<<per;

   }

};

class circle :public shape

{

    private:int radius=5;

    public:

    void area(){

        float area=3.14\*radius\*radius;

        cout<<"area="<<area;

    }

   void perimeter(){

    float per=2\*3.14\*radius;

    cout<<"\nperimeter="<<per;

   }};

   class triangle :public shape

{

    private:int breadth=10,height=4;

    public:

    void area(){

        int area=(height\*breadth)/2;

        cout<<"area="<<area;

    }

   void perimeter(){

    int per=breadth+breadth+height;

    cout<<"\nperimeter="<<per;

   }};

int main(){

  rectangle r;

  r.area();

  r.perimeter();

  circle c;

  c.area();

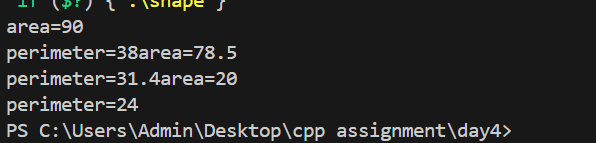
  c.perimeter();

  triangle t;

  t.area();

  t.perimeter();

}



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.

#include<iostream>

using namespace std;

class employee{

    protected:

    string name;

    int empid;

    public:

    employee(){

        name="pratibha";

        empid=234;

    }

    employee(string name,int empid){

        this->name=name;

        this->empid=empid;

    }

    void display(){

        cout<<" emp name: "<<name<<" empid: "<<empid;

    }

    virtual void calsalary()=0;

};

class hourlyemp:public employee{

    int hoursofwork;

    int payperhour;

    public:

    hourlyemp(){

        hoursofwork=8;

        payperhour=200;

    }

    hourlyemp(string name,int empid,int hoursofwork,int payperhur):employee(name,empid){

      this->hoursofwork=hoursofwork;

      this->payperhour=payperhour;

    }

    void display(){

        employee::display();

        cout<<" hoursofwork :"<<hoursofwork<<" payperhour: "<<payperhour<<endl;

    }

    void calsalary(){

       double salary=hoursofwork\*payperhour;

       cout<<"salaryof horlyemp :" <<salary;

    }

};

class salariedemp:public employee{

    private:

    int workdays;

    int payperday;

public:

     salariedemp(){

     workdays=30;

     payperday=2000;

     }

     salariedemp(string name,int empid,int workdays,int payperday):employee(name,empid){

        this->payperday=payperday;

        this->workdays=workdays;

     }

void display()

{

    employee::display();

    cout<<"payperday: "<<payperday;

    cout<<"worksday: "<<workdays;

}

void calsalary(){

    double salary=payperday\*workdays;

    cout<<" salary of salariedemp"<<salary;

}

    };

int main(){

hourlyemp he;

he.display();

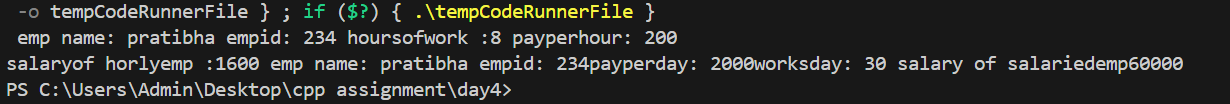
he.calsalary();

salariedemp se;

se.display();

se.calsalary();

}



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().

#include<iostream>

using namespace std;

class vehicle{

   public: virtual void typeoffuel(){

        cout<<"enter the fuel type according to your vehicle\n";

    }

};

class car :public vehicle{

    private :

    int wheelno;

    string name;

    public:

    car(){

        wheelno=4;

        name="porsche";

    }

void typeoffuel(){

    cout<<" car uses cng \n";

}

};

class motercycle:public vehicle{

    private:

    int no\_wheels;

    string name;

    public:

     motercycle(){

        no\_wheels=2;

        name="java";

     }

void typeoffuel(){

    cout<<" motercycle use petrol\n";

}

};

class truck:public vehicle{

    private:

    int no\_wheels;

    public:

    string name;

    truck(){

        no\_wheels=6;

        name="eicheor";

    }

    void typeoffuel(){

        cout<<" truck uses Diesel\n";

    }

};

int main(){

vehicle v;

v.typeoffuel();

car c;

c.typeoffuel();

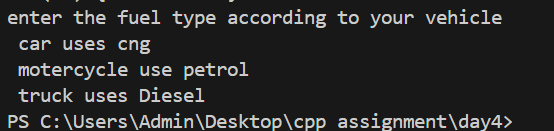
motercycle m;

m.typeoffuel();

truck t;

t.typeoffuel();

}



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 bank\_acc{

    private:

    int acc\_no;

    string name;

    double balance;

    double amt;

    public:

        bank\_acc()

        {

            acc\_no = 0;

            name = "xyz";

            balance = 0;

        }

        bank\_acc(int acc\_no,string name,double balance)

        {

            this->acc\_no = acc\_no;

            this->name = name;

            this->balance = balance;

        }

        void accept (int acc\_no,string name,double balance)

        {

            this->acc\_no = acc\_no;

            this->name = name;

            this->balance = balance;

        }

        void display()

        {

            cout<<"Account no."<<acc\_no<<" Name:"<<name<<" Balance:"<<balance<<endl;

        }

        void withdraw()

        {

            cout<<"\nEnter the amount to withdraw: ";

            cin>>amt;

            cout<<"After withdraw balance is: "<<balance-amt<<endl;

        }

        void deposit()

        {   cout<<"\nEnter the amount to deposit: ";

            cin>>amt;

            cout<<"After deposit balance is: "<<balance+amt<<endl;

        }

};

int main()

{

    double amt;

    bank\_acc b;

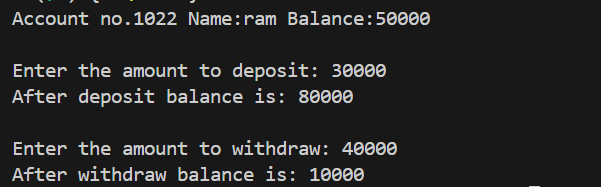
    b.accept(1022,"Ram",50000);

    b.display();

    b.deposit();

    b.withdraw();

}



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{

       void area(){

        cout<<"area of class";

       }

       void perimeter(){

        cout<<"perimeter of class";

       }

};

class rectangle :public shape

{

    public:

    void area(int length,int breadth){

        int area=length\*breadth;

        cout<<"\narea of rectangle= "<<area;

    }

   int perimeter(int length ,int breadth){

    int per=2\*(length+breadth);

    cout<<"\nperimeter of rectangle= "<<per;

   }

};

class circle :public shape

{

    public:

    void area(int radius){

        float area=3.14\*radius\*radius;

        cout<<"\narea of circle= "<<area;

    }

   void perimeter(int radius){

    float per=2\*3.14\*radius;

    cout<<"\nperimeter of circle = "<<per;

   }};

   class triangle :public shape

{

    private:int breadth=10,height=4;

    public:

    void area(){

        int area=(height\*breadth)/2;

        cout<<"\narea of triangle= "<<area;

    }

   void perimeter(){

    int per=breadth+breadth+height;

    cout<<"\nperimeter of triangle= "<<per;

   }};

int main(){

circle c;

c.area(5);

c.perimeter(5);

rectangle r;

r.area(2,5);

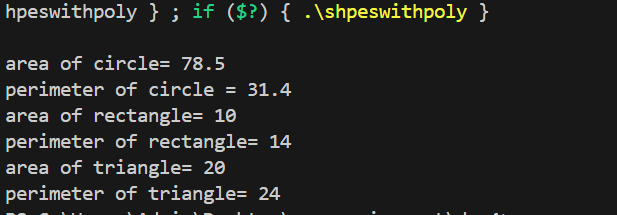
r.perimeter(2,5);

triangle t;

t.area();

t.perimeter();

}



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.

#include<iostream>

using namespace std;

class person{

private:

string name;

int age;

public:

person(){

    name="pratibha";

    age=24;

}

person(string name,int age){

    this->name=name;

    this->age=age;

}

void display(){

    cout<<" \nname: "<<name;

    cout<<" age: "<<age;

}

};

class student:public person{

private:

int stud\_id;

float gpa;

public:

student(){

    stud\_id=2;

    gpa=9.09;

}

student(string name,int age,int stud\_id,float gpa):person(name,age){

    this->stud\_id=stud\_id;

    this->gpa=gpa;

}

void display(){

    person::display();

    cout<<"\n stud\_id: "<<stud\_id<<" gpa: "<<gpa;

}

};

int main(){

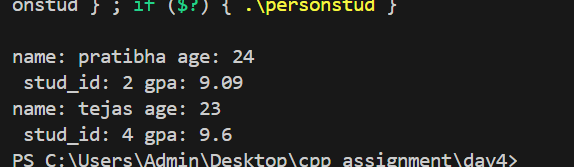
student s;

s.display();

student s1("tejas",23,4,9.60);

s1.display();

}



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{

    protected:

string title;

string auther;

public:

LibraryItem(){

    title="audiobooks";

    auther="rs agrawal";

}

LibraryItem(string title,string auther){

    this->title=title;

    this->auther=auther;

}

void display(){

    cout<<" title :"<<title;

    cout<<" auther :"<<auther;

}

};

class book:public LibraryItem{

    private:

           string bookname;

           string booktype;

           int pages;

         public:

           book(){

            bookname="bhagvadgita";

            booktype="spiritual";   //political,thriller ,romance ,novels;

            pages=300;

           }

           book(string title,string auther,string bookname,string booktype,int pages):LibraryItem(title,auther){

            this->bookname=bookname;

            this->booktype=booktype;

            this->pages=pages;

           }

           void display(){

            LibraryItem::display();

            cout<<" bookname :"<<bookname<<" booktype:"<<booktype<<" pages :"<<pages<<endl;

           }

};

class journals :public LibraryItem{

    private:

    string topic;

   int  duration\_of\_publication;

   int team\_members;

public:

journals(){

    topic="bio";

    duration\_of\_publication=9;

    team\_members=2;

}

journals(string title,string auther,string topic,int  duration\_of\_publication,int team\_members):LibraryItem(title,auther)

{

    this->topic=topic;

    this->duration\_of\_publication=duration\_of\_publication;

    this->team\_members=team\_members;

}

void display(){

    LibraryItem::display();

    cout<<"topic: "<<topic<<" duration\_of\_publication"<<duration\_of\_publication<<" team\_membaers:"<<team\_members<<endl;

}

};

int main(){

book b;

b.display();

book b1("bahgvadgta","krishna","shreemadbhagvatgita","spritual",500);

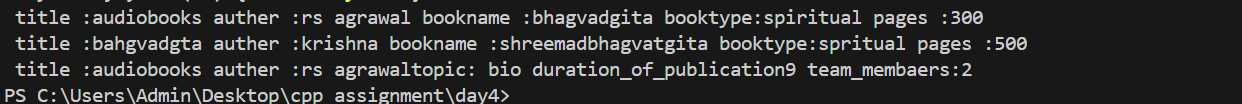
b1.display();

journals j;

j.display();

journals j1("maths","arybhatta","integration",5,3);

}



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.

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{

    protected:

    string name;

    int empid;

    public:

    employee(){

        name="pratibha";

        empid=234;

    }

    employee(string name,int empid){

        this->name=name;

        this->empid=empid;

    }

    void display(){

        cout<<" emp name: "<<name<<" empid: "<<empid;

    }

    virtual void calsalary()=0;

};

class hourlyemp:public employee{

    int hoursofwork;

    int payperhour;

    public:

    hourlyemp(){

        hoursofwork=8;

        payperhour=200;

    }

    hourlyemp(string name,int empid,int hoursofwork,int payperhur):employee(name,empid){

      this->hoursofwork=hoursofwork;

      this->payperhour=payperhour;

    }

    void display(){

        employee::display();

        cout<<" hoursofwork :"<<hoursofwork<<" payperhour: "<<payperhour<<endl;

    }

    void calsalary(){

       double salary=hoursofwork\*payperhour;

       cout<<"salaryof horlyemp :" <<salary;

    }

};

class salariedemp:public employee{

    private:

    int workdays;

    int payperday;

public:

     salariedemp(){

     workdays=30;

     payperday=2000;

     }

     salariedemp(string name,int empid,int workdays,int payperday):employee(name,empid){

        this->payperday=payperday;

        this->workdays=workdays;

     }

void display()

{

    employee::display();

    cout<<"payperday: "<<payperday;

    cout<<"worksday: "<<workdays;

}

void calsalary(){

    double salary=payperday\*workdays;

    cout<<" salary of salariedemp"<<salary;

}

    };

int main(){

hourlyemp he;

he.display();

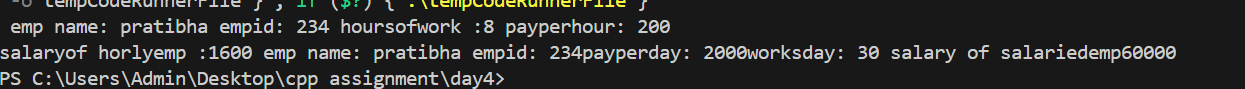
he.calsalary();

salariedemp se;

se.display();

se.calsalary();

}



Day 5

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

----------------------------------------------------

#include<iostream>

using namespace std;

class employee{

    private:

        int id;

        string name;

        int depId;

    protected: double salary;

    public:

        employee()

        {

            id=1;

            name=" Siya";

            depId = 0;

            salary= 60000;

        }

        employee(int id,string name,int depId,double salary)

        {

            this->id=id;

            this->name=name;

            this->depId=depId;

            this->salary=salary;

        }

        void accept(int id,string name,int depId,double salary)

        {

            this->id=id;

            this->name=name;

            this->depId=depId;

            this->salary=salary;

        }

        void display()

        {

            cout<<"ID="<<id<<"  Name="<<name<<" Department ID="<<depId<<"  salary="<<salary<<endl;

        }

        virtual void calSalary()

        {

            cout<<"Employee Salary= "<<salary<<endl;

        }

};

class manager:public employee

{

    private: double bonus;

    public:

        manager()

        {

            this->bonus=5000;

        }

        manager(int id,string name,int deptId,double salary,double bonus)

        :employee(id,name,deptId,salary)

        {

            this->bonus=bonus;

        }

        void accept(double bonus)

        {

            this->bonus=bonus;

        }

        void manager\_task()

        {

            cout<<"\n\n---Manager task---"<<endl;

        }

        void calSalary()

        {

            cout<<"Manager Salary= "<<salary+bonus<<endl;

        }

        void display()

        {

            employee::display();

            cout<<"  Bonus="<<bonus<<endl;

        }

};

class worker:public employee

{

    private: int hr\_worked;

        int hr\_rate;

    public:

        worker()

        {

            this->hr\_worked=0;

            this->hr\_rate = 0;

        }

        worker(int id,string name,int deptId,double salary,int hr\_worked, int hr\_rate)

        :employee(id,name,deptId,salary)

        {

            this->hr\_worked=hr\_worked;

            this->hr\_rate = hr\_rate;

        }

        void accept(int hr\_worked, int hr\_rate)

        {

            this->hr\_worked=hr\_worked;

            this->hr\_rate = hr\_rate;

        }

        void workers\_task()

        {

            cout<<"\n\n---workers task---"<<endl;

        }

        void calSalary()

        {

            cout<<"Workers Salary= "<<salary+(hr\_worked\*hr\_rate)<<endl;

        }

        void display()

        {

            employee::display();

            cout<<"  Working hours="<<hr\_worked<<" Pay per hours="<<hr\_rate<<endl;

        }

};

int main()

{

    employee emp;

    manager mngr;

    worker w;

    int ch;

    do{

        cout<<"\n\nEnter choice:";

        cin>>ch;

        switch(ch){

            case 1:

                cout<<"Hire Manager"<<endl;

                //manager mngr;

                mngr.accept(10000);

                mngr.manager\_task();

                mngr.display();

                mngr.calSalary();

                break;

            case 2:

                cout<<"Hire workers"<<endl;

                //worker w;

                w.accept(8,100);

                w.workers\_task();

                w.display();

                w.calSalary();

                break;

            case 3:

                cout<<"\n\nUsing array"<<endl;

                int i;

                employee\* pay[3];

                pay[0]=&mngr;

                pay[1]=&w;

                for(i=0;i<3;i++)

                {

                    pay[i]->display();

                    pay[i]->calSalary();

                }

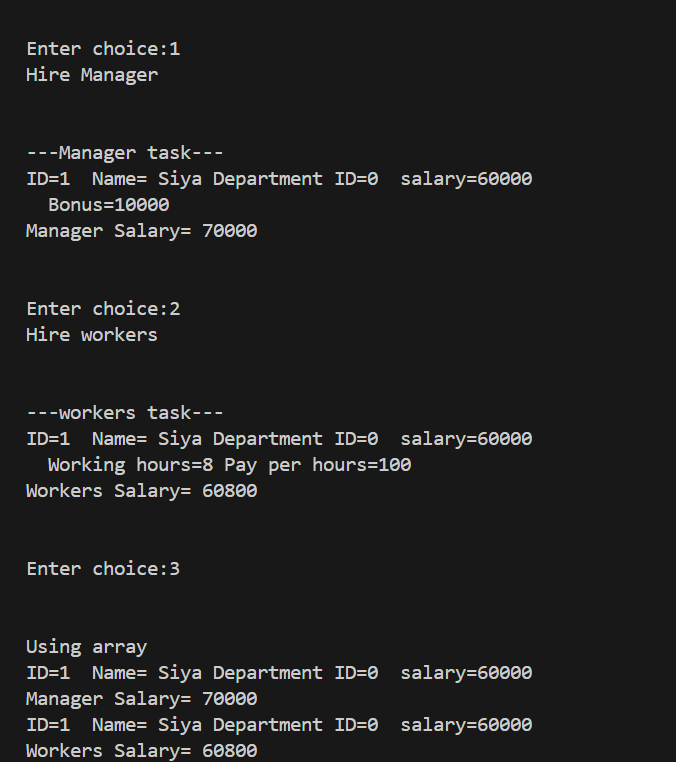
        }

}

        while(ch!=4);

        return 0;

}



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.

---------------------------------------------------------------------------------

#include<iostream>

using namespace std;

class bank\_acc{

    private:

    int acc\_no;

    string name;

    double balance;

    double amt;

    public:

        bank\_acc()

        {

            acc\_no = 0;

            name = "xyz";

            balance = 0;

        }

        bank\_acc(int acc\_no,string name,double balance)

        {

            this->acc\_no = acc\_no;

            this->name = name;

            this->balance = balance;

        }

        void accept (int acc\_no,string name,double balance)

        {

            this->acc\_no = acc\_no;

            this->name = name;

            this->balance = balance;

        }

        void display()

        {

            cout<<"Account no."<<acc\_no<<" Name:"<<name<<" Balance:"<<balance<<endl;

        }

        void withdraw()

        {

            cout<<"\nEnter the amount to withdraw: ";

            cin>>amt;

            cout<<"After withdraw balance is: "<<balance-amt<<endl;

        }

        void deposit()

        {   cout<<"\nEnter the amount to deposit: ";

            cin>>amt;

            cout<<"After deposit balance is: "<<balance+amt<<endl;

        }

};

int main()

{

    double amt;

    bank\_acc b;

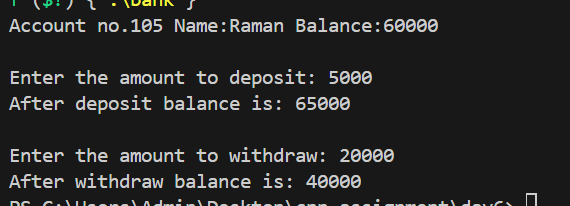
    b.accept(105,"Raman",60000);

    b.display();

    b.deposit();

    b.withdraw();

}



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.

#include<iostream>

using namespace std;

class shape{

    protected: float area;

    public:

    shape()

    {

        area=0;

    }

    virtual void cal\_area()=0;

};

class circle:public shape{

    private: int radius;

    public:

    circle()

    {

        radius = 0;

    }

    circle(int radius)

    {

        this->radius=radius;

    }

    void accept(int radius)

    {

        this->radius=radius;

    }

    void cal\_area(){

        area = 3.142\*radius\*radius;

    }

    void display()

    {

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

    }

};

class rectangle:public shape{

    private: float length;

        float breadth;

    public:

    rectangle()

    {

        length = 0;

        breadth = 0;

    }

    rectangle(float length,float breadth)

    {

        this->length=length;

        this->breadth=breadth;

    }

    void accept(float length,float breadth)

    {

        this->length=length;

        this->breadth=breadth;

    }

    void cal\_area(){

        area = length\*breadth;

    }

    void display()

    {

        cout<<"Area of the rectangle is :"<<area<<endl;

    }

};

class square:public shape{

    private:

        float side1;

    public:

    square()

    {

        side1=0;

    }

    square(float side1)

    {

        this->side1=side1;

    }

    void accept(float side1)

    {

        this->side1=side1;

    }

    void cal\_area(){

        area = side1\*side1;

    }

    void display()

    {

        cout<<"Area of the square is :"<<area<<endl;

    }

};

int main()

{

    cout<<"\nAbout Circle:"<<endl;

    circle crl;

    crl.accept(5);

    crl.cal\_area();

    crl.display();

    cout<<"\nAbout rectangle:"<<endl;

    rectangle rect;

    rect.accept(2,3);

    rect.cal\_area();

    rect.display();

    cout<<"\n About square:"<<endl;

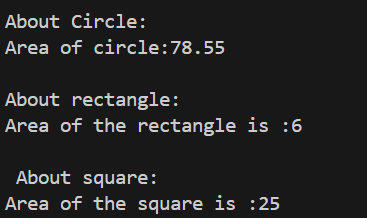
    square sq;

    sq.accept(5);

    sq.cal\_area();

    sq.display();

}



Day 6

assignment: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

s.pop();//

#include<iostream>

using namespace std;

class stack

{

    private:

    int a[4];

    int top;

    public: stack()

        {

                this->a[4]={-1};

                top=-1;

        }

        void push(int value)

{

      if(top==3)

        {

           // cout<<"Stack is full\n"<<endl;

           throw 10;

        }

        else

        {

           cout<<"Add element on stack\n"<<endl;

           top++;

           a[top]=value;

        }

}

        void display()

        {

            for(int i=top;i>=0;--i)

            {

                cout<<a[i]<<endl;

            }

    }

     void pop()

        {

            if(top==-1)

            {

               // cout<<"Stack is empty\n"<<endl;

               throw 20;

            }

            else{

           cout<<"Remove element on stack\n"<<endl;

           a[top]=-1;

           top--;

            }

        }

        void seek()

        {    cout<<"Displaying top element from stack is\n"<<endl;

           cout<<a[top]<<endl;;

        }

};

int main()

{

    stack s;

   try

   {

    s.push(10);

    s.display();

    s.push(20);

    s.display();

    s.push(30);

    s.display();

    s.seek();

   s.push(40);

    s.display();

    s.push(10);

    s.display();

   }

    catch(int no)

    {

        if (no==10)

        cout<<"stack is full"<<endl;

        else

        cout<<"stack is empty"<<endl;

    }

    return 0;

}

