**Day1:**

**1:write program to test Hello World.**

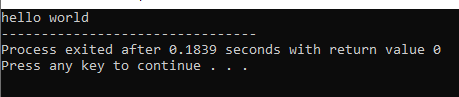
**#include<iostream>**

**using namespace std;**

**int main1(){**

**cout<<"hello world";**

**}**

****

**2:Write a program to adddition of two numbers .**

**#include<iostream>**

**using namespace std;**

**int main2(){**

**int no1,no2,result;**

**cout<<"enter no1:";**

**cin>>no1;**

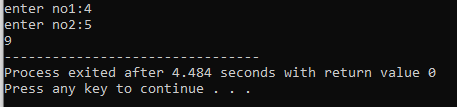
**cout<<"enter no2:";**

**cin>>no2;**

**result=no1+no2;**

**cout<<result;**

**}**

****

**3:Write a program to swap two numbers.**

**#include<iostream>**

**using namespace std;**

**int main3(){**

**int a,b,temp;**

**cout<<"enter a:";**

**cin>>a;**

**cout<<"enter b:";**

**cin>>b;**

**cout<<"before swap"<<"a: "<<a<<" b: "<<b<<endl;**

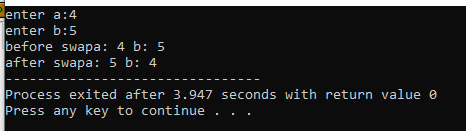
**temp=b;**

**b=a;**

**a=temp;**

**cout<<"after swap"<<"a: "<<a<<" b: "<<b;**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main4(){**

**int no;**

**cout<<"enter no: ";**

**cin>>no;**

**if(no%2==0){**

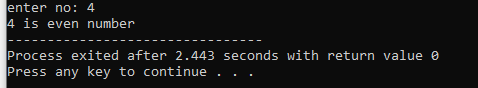
**cout<<no<<" is even number";**

**}else{**

**cout<<no<<" is odd number";**

**}**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main5(){**

**int no;**

**cout<<"enter no: ";**

**cin>>no;**

**if(no%5==0 && no%7==0){**

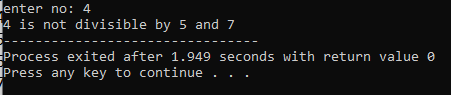
**cout<<no<<" is divisible by 5 and 7";**

**}else{**

**cout<<no<<" is not divisible by 5 and 7";**

**}**

**}**

****

**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 main6(){**

**int salary,tax;**

**cout<<"enter the salary";**

**cin>>salary;**

**if(salary<=150000){**

**cout<<"tax is 0";**

**}else if(salary>15000&&salary<=300000){**

**tax=0.2\*(salary-150000);**

**cout<<"tax is "<<tax;**

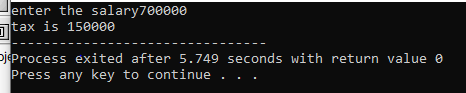
**}else{**

**tax=0.2\*(300000-150000)+0.3\*(salary-300000);**

**cout<<"tax is "<<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 main7(){**

**char ch;**

**cout<<"enter the charecter ch: ";**

**cin>>ch;**

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

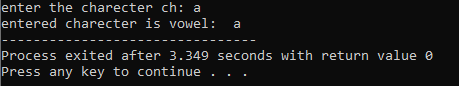
**cout<<"entered charecter is vowel: "<<ch;**

**}else {**

**cout<<"entered charecter is consonant: "<<ch;**

**}**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main8(){**

**int angle1,angle2,angle3;**

**cout<<"enter angle1,angle2,angle3: "<<endl;**

**cin>>angle1>>angle2>>angle3;**

**if((angle1+angle2+angle3)==180){**

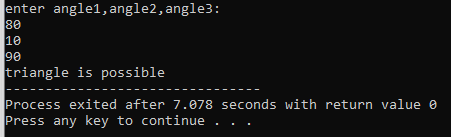
**cout<<"triangle is possible";**

**}else {**

**cout<<"triangle is not possible";**

**}**

**}**

****

**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 main9(){**

**int no,fact=1;**

**cout<<"enter the no for which you want factorial";**

**cin>>no;**

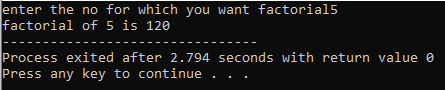
**for(int i=1;i<=no;i++){**

**fact=fact\*i;**

**}**

**cout<<"factorial of "<<no<<" is "<<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 main10(){**

**int base,pow,ans=1;**

**cout<<"enter base and power";**

**cin>>base>>pow;**

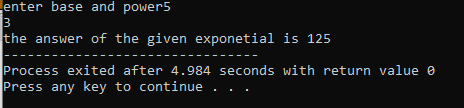
**for(int i=1;i<=pow;i++){**

**ans=ans\*base;**

**}**

**cout<<"the answer of the given exponetial is "<<ans;**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main11(){**

**int num;**

**bool flag=false;**

**cout<<"enter no: ";**

**cin>>num;**

**if(num==0||num==1){**

**cout<<"not prime";**

**}else{**

**for(int i=2;i<num;i++){**

**if(num%i==0){**

**flag=true;**

**break;**

**}**

**}**

**if(flag){**

**cout<<"not prime";**

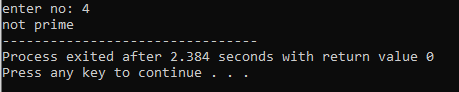
**}else{**

**cout<<"prime";**

**}**

**}**

**}**

****

**12:Sum of series :**

**1+2+3+….+n**

**#include<iostream>**

**using namespace std;**

**int main12(){**

**int n;**

**int sum=0;**

**cout<<"enter n";**

**cin>>n;**

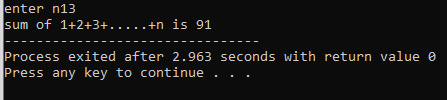
**for(int i=0;i<=n;i++){**

**sum=sum+i;**

**}**

**cout<<"sum of 1+2+3+.....+n is "<<sum;**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main13(){**

**int num,rem,rev=0,num1;**

**cout<<"enter number: ";**

**cin>>num;**

**num1=num;**

**for(int i=0;num%10>0;i++){**

**rem=num%10;**

**num=num/10;**

**rev=rev\*10+rem;**

**}**

**if(rev==num1){**

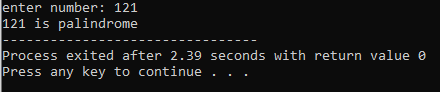
**cout<<num1<<" is palindrome";**

**}else{**

**cout<<num1<<" is not palindrome";**

**}**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main14(){**

**int n,sum\_even,sum\_odd;**

**cout<<"enter n:";**

**cin>>n;**

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

**if(i%2==0){**

**sum\_even=sum\_even+i;**

**}else{**

**sum\_odd=sum\_odd+i;**

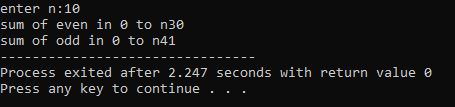
**}**

**}**

**cout<<"sum of even in 0 to n"<<sum\_even<<endl;**

**cout<<"sum of odd in 0 to n"<<sum\_odd;**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main15(){**

**int n,rev=0;**

**cout<<"enter n: ";**

**cin>>n;**

**for(int i=0;n%10>0;i++){**

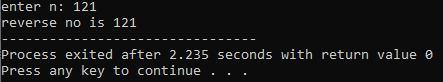
**rev=(rev\*10)+(n%10);**

**n=n/10;**

**}**

**cout<<"reverse no is "<<rev;**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main16(){**

**int n;**

**bool flag=false;**

**cout<<"enter n: ";**

**cin>>n;**

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

**for(int j=2;j<i;j++){**

**if(i%j==0){**

**flag=true;**

**}**

**}**

**if(flag==false){**

**cout<<i<<" ";**

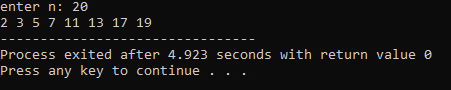
**}else{**

**flag=false;**

**}**

**}**

**}**

****

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

**#include<iostream>**

**#include<cmath>**

**using namespace std;**

**int main(){**

**int n,remainder,count=0,sum=0;**

**cout<<"enter no: ";**

**cin>>n;**

**int n2=n;**

**for(int i=0;n%10>0;i++){**

**count++ ;**

**n=n/10;**

**}**

**n=n2;**

**for(int i=0;i<count;i++){**

**remainder=n%10;**

**n=n/10;**

**sum=sum+(pow(remainder,count));**

**}**

**if(sum==n2){**

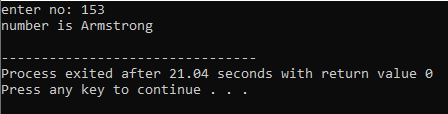
**cout<<"number is Armstrong"<<endl;**

**}else{**

**cout<<"number is not Armstrong"<<endl;**

**}**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main18(){**

**int a,b,c;**

**cout<<"enter a,b,c: ";**

**cin>>a>>b>>c;**

**if((a>b)&&(a>c)){**

**cout<<"a is greatest number";**

**}else if((b>a)&&(b>c)){**

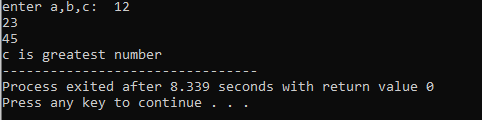
**cout<<"b is greatest number";**

**}else{**

**cout<<"c is greatest number";**

**}**

**}**

****

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

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int total=0,ch;**

**do**

**{**

**cout<<"====================="<<endl;**

**cout<<"1. Pizza F1"<<endl;**

**cout<<"2. Pizza F2"<<endl;**

**cout<<"3. Cola"<<endl;**

**cout<<"4. Fries"<<endl;**

**cout<<"5. Total Bill"<<endl;**

**cout<<"0. Exit"<<endl;**

**cout<<"====================="<<endl;**

**cout<<"Enter Choice : ";**

**cin>>ch;**

**cout<<"====================="<<endl;**

**switch(ch)**

**{**

**case 1:{**

**total+=150;**

**cout<<"Pizza F1 added"<<endl;**

**break;**

**}**

**case 2:{**

**total+=160;**

**cout<<"Pizza F2 added"<<endl;**

**break;**

**}**

**case 3:{**

**total+=25;**

**cout<<"Cola added"<<endl;**

**break;**

**}**

**case 4:{**

**total+=60;**

**cout<<"Fries added"<<endl;**

**break;**

**}**

**case 5:{**

**cout<<"Total Bill Rs."<<total<<endl;**

**break;**

**}**

**case 0:{**

**cout<<"Exiting..!!!!";**

**return 0;**

**break;**

**}**

**default:{**

**cout<<"Enter Correct Option"<<endl;**

**break;**

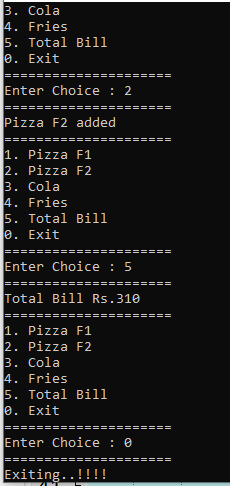
**}**

**}**

**}while(ch!=0);**

**return 0;**

**}**

****

**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 choice;**

**cout<<"enter chice: ";**

**cin>>choice;**

**switch(choice){**

**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;**

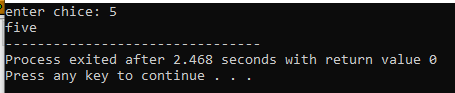
**case 8:cout<<"eight";break;**

**case 9:cout<<"nine";break;**

**default:break;**

**}**

**}**

****

**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(){

double no1,no2,sum,sub,mul,div;

char ch;

if(ch=='+'){

sum=no1+no2;

cout<<"sum:"<<sum;

}else if(ch=='-'){

sub=no1-no2;

cout<<"substraction:"<<sub;

}else if(ch=='\*'){

mul=no1\*no2;

cout<<"multiplication:"<<mul;

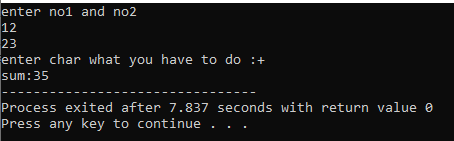
}else{

div=no1/no2;

cout<<"division:"<<div;

}

}



**Day 2:**

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;

do{

cout<<"enter the positive number: ";

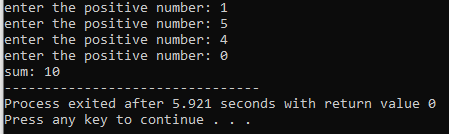
cin>>n;

sum=sum+n;

}while(n>0);

cout<<"sum: "<<sum;

}



2. 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,ans=1;

cout<<"enter value of x: ";

cin>>x;

cout<<"Enter value of n: ";

cin>>n;

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

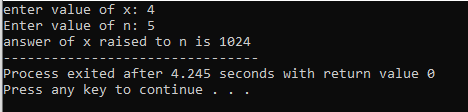
{

ans = ans \*x;

}

cout<<"answer of x raised to n is "<<ans;

}



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

#include<iostream>

using namespace std;

int main(){

int n;

char ch;

cout<<"enter n:";

cin>>n;

cout<<"enter ch:";

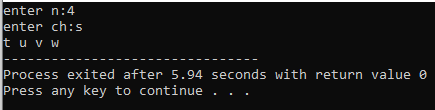
cin>>ch;

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

cout<<char(ch+i)<<" ";

}

}



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 num,fact=1;

cout<<"Enter a number: ";

cin>>num;

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

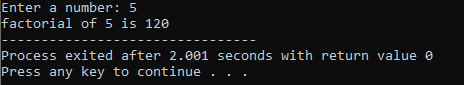
{

fact=fact\*i;

}

cout<<"factorial of "<<num<<" is "<<fact;

}



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

#include<iostream>

using namespace std;

int main(){

int num;

cout<<"enter num: "<<endl;

cin>>num;

cout<<"factors of "<<num<<" is ";

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

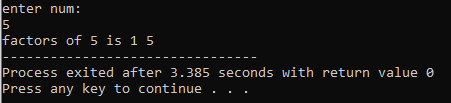
if(num%i==0){

cout<<i<<" ";

}

}

}



6. Accept two numbers and calculate GCD of them.

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

int num1,num2,temp;

cout<<"Enter two numbers to find GCD: ";

cin>>num1>>num2;

for(int i=1;i<=max(num1,num2);i++)

{

if(num1%i==0&&num2%i==0)

{

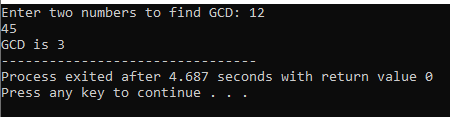
temp=i;

}

}

cout<<"GCD is "<<temp;

}



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()

{

double radius,length,breadth,height,base,areacir,arearec,areatri;

int choice;

cout<<"1-area of circle 2-area of rectangle 3-area of triangle 4-exit"<<endl;

do{

cout<<"enter your choice";

cin>>choice;

switch (choice){

case 1:

cout<<"enter the radius: ";

cin>>radius;

areacir=3.1415\*radius\*radius;

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

break;

case 2:

cout<<"enter the length and breath : ";

cin>>length>>breadth;

arearec=length\*breadth;

cout<<"area of rectangle: "<<arearec<<endl;

break;

case 3:

cout<<"enter the height and base : ";

cin>>height>>base;

areatri=0.5\*height\*base;

cout<<"area of triangle: "<<areatri<<endl;

break;

case 4:cout<<"exit";

break;

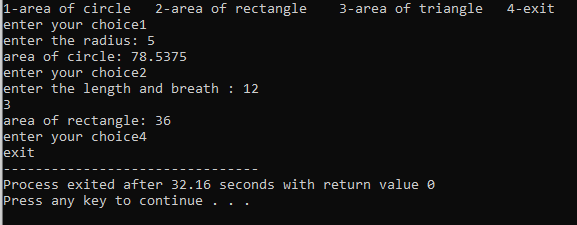
default:cout<<"invalid";

break;

};

}while(choice!=4);

}



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

#include<iostream>

using namespace std;

int main()

{

int num;

bool flag= false;

cout<<"Enter a number: ";

cin>>num;

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

{

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

{

if(i%j==0)

{

flag = true;

}

}

if(flag==false)

{

cout<<i<<" ";

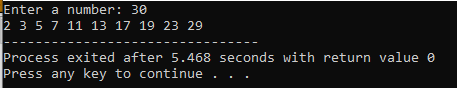
}

else

flag=false;

}

}



**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.**

**#include<iostream>**

**using namespace std;**

**class student{**

**private:int rollno;**

**int marks1;**

**int marks2;**

**int marks3;**

**public:void acceptInfo(){**

**cout<<"enter rollno: ";**

**cin>>rollno;**

**cout<<"enter marks1: ";**

**cin>>marks1;**

**cout<<"enter marks2: ";**

**cin>>marks2;**

**cout<<"enter marks3: ";**

**cin>>marks3;**

**}**

**public:void display(){**

**cout<<"roll no:"<<rollno<<" marks1:"<<marks1<<" marks2:"<<marks2<<" marks3:"<<marks3<<endl;**

**}**

**public:void displaycalculation(){**

**int total=marks1+marks2+marks3;**

**cout<<"total: "<<total<<endl;**

**double percentage=total/3.0;**

**cout<<"percentage: "<<percentage<<endl;**

**if(percentage>80){**

**cout<<"grade= a";**

**}else if(percentage>60){**

**cout<<"grade= b";**

**}**

**else if(percentage>40)**

**{**

**cout<<"grade= c";**

**}**

**else**

**cout<<"grade= f";**

**}**

**};**

**int main(){**

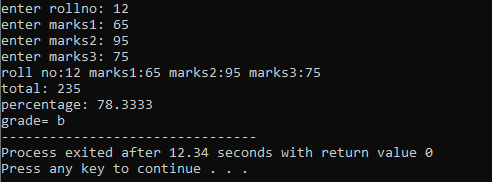
**student s1;**

**s1.acceptInfo();**

**s1.display();**

**s1.displaycalculation();**

**}**

****

**1. Create a class Person with data members as name, age, city.**

**members. Also add accept and 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 Person{**

**private:string name;**

**int age;**

**string city;**

**public:void accept()**

**{**

**cout<<"Enter name: ";**

**cin>>name;**

**cout<<"Enter age: ";**

**cin>>age;**

**cout<<"Enter city: ";**

**cin>>city;**

**}**

**public:void display()**

**{**

**cout<<"Your name is: "<<name<<endl;**

**cout<<"Your age is: "<<age<<endl;**

**cout<<"Your city is: "<<city<<endl;**

**}**

**};**

**int main()**

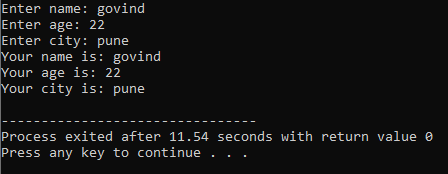
**{**

**Person p1;**

**p1.accept();**

**p1.display();**

**}**

****

**2. Create a class Date with data members as dd, mm, yy. Create AcceptDate function. 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 Date{**

**private:int dd;**

**int mm;**

**int yyyy;**

**public:void AcceptDate(){**

**cout<<"enter day between 1 to 31: ";**

**cin>>dd;**

**cout<<"enter month between 1 to 12**

**: ";**

**cin>>mm;**

**cout<<"enter year: ";**

**cin>>yyyy;**

**}**

**public:void display(){**

**cout<<"dd:mm:yyyy="<<dd<<":"<<mm<<":"<<yyyy<<endl;**

**}**

**};**

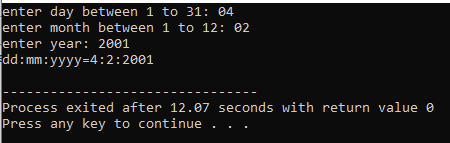
**int main(){**

**Date d1;**

**d1.AcceptDate();**

**d1.display();**

**}**

****

**3. Create a class Book with data members as bname,id,author,price. Write AcceptBook function . 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;**

**float price;**

**public:void AcceptBook(){**

**cout<<"Enter book name: ";**

**cin>>bname;**

**cout<<"Enter book id: ";**

**cin>>id;**

**cout<<"Enter author: ";**

**cin>>author;**

**cout<<"Enter price: ";**

**cin>>price;**

**}**

**public:void display()**

**{**

**cout<<"Book name: "<<bname<<endl;**

**cout<<"Book id: "<<id<<endl;**

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

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

**}**

**};**

**int main()**

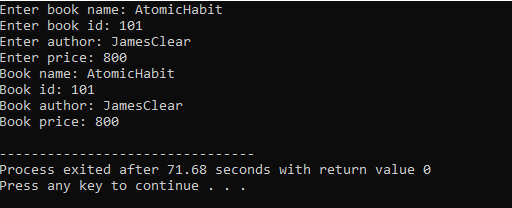
**{**

**Book b1;**

**b1.AcceptBook();**

**b1.display();**

**}**

****

**4. Create a class Point with data members as x,y. Add AcceptPoint and 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:void AcceptPoint(){**

**cout<<"enter x :";**

**cin>>x;**

**cout<<"enter y :";**

**cin>>y;**

**}**

**public:void display(){**

**cout<<"value of x:"<<x<<" value of y:"<<y<<endl;**

**}**

**};**

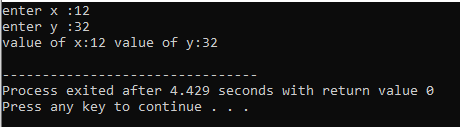
**int main(){**

**Point p1;**

**p1.AcceptPoint();**

**p1.display();**

**}**

****

**5. Create a class ComplexNumber with data members real, imaginary. Create AcceptComplexNumber() and 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 imaginary;

public:void AcceptComplexNumber(){

cout<<"enter real part of complex no: ";

cin>>real;

cout<<"enter imaginary part of complex no: ";

cin>>imaginary;

}

public:void display(){

cout<<"complex number is: "<<real<<"+"<<imaginary<<"i";

}

};

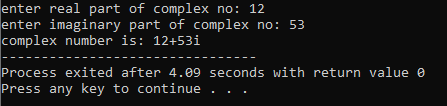
int main(){

ComplexNumber cn1;

cn1.AcceptComplexNumber();

cn1.display();

}



**Day:4**

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,city;

int age;

public: Person(){

name = "Govind";

age = 21;

city = "Nashik";

}

public: Person(string name,int age, string city){

this->name = name;

this->age = age;

this->city = city;

}

//getter

string getName(){

return name;

}

int getAge(){

return age;

}

string getCity(){

return city;

}

//setter

void setName(int name){

this->name = name;

}

void setAge(int age){

this->age = age;

}

void setCity(string city){

this->city = city;

}

public: void Display(){

cout<<"Name is: "<<name<<" "<<"Age is: "<<age<<" City is : "<<city<<endl;

}

};

int main(){

Person pr;

pr.Display();

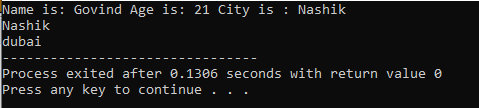
cout<<pr.getCity()<<endl;

pr.setCity("dubai");

cout<<pr.getCity();

return 0;

}



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 date,month,year;

public: Date(){

date=01;

month=05;

year=1900;

}

public: Date(int date,int month,int year){

this->date=date;

this->month=month;

this->year=year;

}

//getter

int getDate(){

return date;

}

int getMonth(){

return month;

}

int getYear(){

return year;

}

//setter

void setDate(int date){

this->date=date;

}

void setMonth(int month){

this->month=month;

}

void setYear(int year){

this->year=year;

}

//display function

public:void display(){

cout<<"dd-mm-yyyy = "<<date<<"-"<<month<<"-"<<year<<endl;

}

};

int main(){

Date d1;

d1.display();

Date d2;

d2.setDate(04);

d2.setMonth(02);

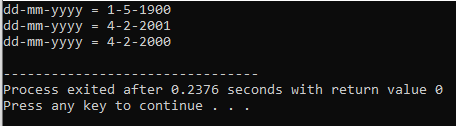
d2.setYear(2001);

d2.display();

d2.setYear(2000);

d2.display();

}



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 bookName, author;

int book\_Id, book\_Price;

public: Book(){

bookName = "Ikigai";

author = "Govind ";

book\_Id = 121;

book\_Price = 239;

}

public: Book(string bookName, string author, int book\_Id, int book\_Price){

this->bookName = bookName;

this->author = author;

this->book\_Id = book\_Id;

this->book\_Price = book\_Price;

}

// Getter

string getBookName(){

return bookName;

}

string getAuthor(){

return author;

}

int getBook\_Id(){

return book\_Id;

}

int getBook\_Price(){

return book\_Price;

}

//Setters

void setBookName(string bookName){

this->bookName = bookName;

}

void setAuthor(string author){

this->author = author;

}

void setBook\_Id(int book\_Id){

this->book\_Id = book\_Id;

}

void setBook\_Price(int book\_Price){

this->book\_Price = book\_Price;

}

public: void display(){

cout<<"Name of the Book is : "<<bookName<<" and author is : "<<author<<" book id of the book is: "<<book\_Id<<" and Price of the book is: "<<book\_Price<<endl;

}

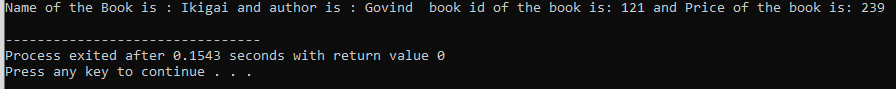
};

int main(){

Book bk;

bk.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,y;

public:Point(){

x=10;

y=10;

}

public:Point(int x,int y){

this->x=x;

this->y=y;

}

//getter

public:int getX(){

return x;

}

public:int getY(){

return y;

}

//setter

public:void setX(int x){

this->x=x;

}

public :void setY(int y){

this->y=y;

}

//display

public:void display(){

cout<<"x:"<<x<<" y:"<<y<<endl;

}

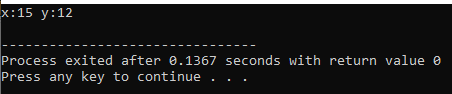
};

int main(){

Point p1(15,12);

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 ComplexNumber{

private:int real,imaginary;

public:ComplexNumber(){

real=10;

imaginary=20;

}

public:ComplexNumber(int real,int imaginary){

this->real=real;

this->imaginary=imaginary;

}

//getter

public:int getReal(){

return real;

}

public:int getImaginary(){

return imaginary;

}

//setter

public:void setReal(int real){

this->real=real;

}

public:void setImaginary(int imaginary){

this->imaginary=imaginary;

}

//display function

public: void display(){

cout<<"complex number"<<real<<"+"<<imaginary<<"i"<<endl;

}

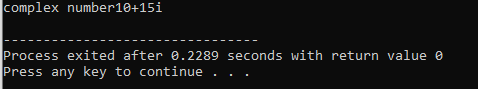
};

int main(){

ComplexNumber cn(10,15);

cn.display();

}



**Day:5**

**1:Create Date class with members day,month ,year.**

**Write no argument and parameterised constructor .Create two object s and initialize them using no argument and parameterised constructor**

**respectively.Print date using display function.**

**#include<iostream>**

**using namespace std;**

**class Date{**

**private:int day,month,year;**

**//constructor**

**public: Date(){**

**day=01;**

**month=06;**

**year=2001;**

**}**

**Date(int day,int month,int year){**

**this->day=day;**

**this->month=month;**

**this->year=year;**

**}**

**//function**

**public:void display(){**

**cout<<"day: "<<day<<" month: "<<month<<" year: "<<year<<endl;**

**}**

**};**

**int main(){**

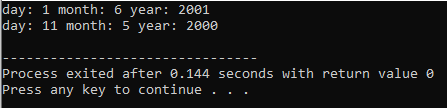
**Date d1;**

**Date d2(11,05,2000);**

**d1.display();**

**d2.display();**

**}**

****

**2:Create Employee class with members id(int),name(string),dob(Date).Use above created Date class.**

**Write default and parameterised constructor in Employee Class.Write accept() function to accept information and display() to display emp information.**

**#include<iostream>**

**using namespace std;**

**class Date{**

**protected:int day,month,year;**

**//constructor**

**public:Date(){**

**day=01;**

**month=06;**

**year=2001;**

**}**

**public:Date(int day,int month,int year){**

**this->day=day;**

**this->month=month;**

**this->year=year;**

**}**

**};**

**class Employee:public Date{**

**private: int id;**

**string name;**

**//Date dob;**

**public: Employee(){**

**id = 101;**

**name = "Govind";**

**}**

**public: Employee(int id, string name,int day,int month,int year):Date(day,month,year){**

**this->id=id;**

**this->name=name;**

**}**

**public: void display(){**

**cout<<id<<" "<<name<<" "<<day<<"-"<<month<<"-"<<year<<endl;**

**}**

**};**

**int main(){**

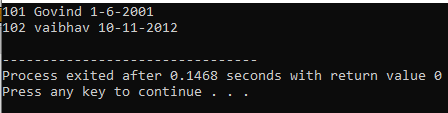
**Employee e1;**

**e1.display();**

**Employee e2(102,"vaibhav",10,11,2012);**

**e2.display();**

**}**

****

**3:Consider that payroll software needs to be developed for computerization of**

**operations of an ABC organization. The organization has employees.**

**3.1. Construct a class Employee with following members using private access**

**specifies:**

**Employee Id integer**

**Employee Name string**

**Basic Salary double**

**HRA double**

**Medical double=1000**

**PF double**

**PT double**

**Net Salary double**

**Gross Salary double**

**Please use following expressions for calculations://Note:Don't accept HRA,PF PT from user**

**\* HRA = 50% of Basic Salary**

**\* PF = 12% of Basic Salary**

**\* PT = Rs. 200**

**3.2. Write methods to display the details of an employee and calculate the gross**

**and net salary.**

**\* Goss Salary = Basic Salary + HRA + Medical**

**\* Net Salary = Gross Salary – (PT + PF)**

**Create Object of employee class and assign values and display Details.**

#include<iostream>

using namespace std;

class Employee{

private:int EmployeeId;

string name;

double basicSalary, hra,pf, netSalary,grossSalary;

double medical=1000;

double pt = 200;

public:Employee(){

basicSalary = 50000;

}

public:Employee(double basicSalary){

this->basicSalary = basicSalary;

}

public: void calculate(){

hra = 0.50\* basicSalary;

pf = 0.12\*basicSalary;

grossSalary = basicSalary + hra + medical;

netSalary = (grossSalary) - (pf + pt);

}

public:void display(){

cout<<"Gross Salary is : "<<grossSalary<<" and net Salary is "<<netSalary;

}

};

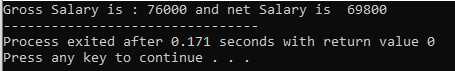
int main(){

Employee e1;

e1.calculate();

e1.display();

}

****

**Day:6**

**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>**

**#include<typeinfo>**

**using namespace std;**

**class Employee{**

**public:int id;**

**string name;**

**int deptId;**

**double basicSalary;**

**//constructor**

**public:Employee(){**

**id=101;**

**name="pushparaj";**

**deptId=143;**

**basicSalary=50000;**

**}**

**public:Employee(int id,string name,double basicSalary,int deptId){**

**this->id=id;**

**this->name=name;**

**this->deptId=deptId;**

**this->basicSalary=basicSalary;**

**}**

**//function**

**public:virtual double computeNetSalary(){**

**cout<<"hhh";**

**return 12;**

**}**

**};**

**class Manager:public Employee{**

**private:**

**int perfBonus;**

**//constructorm**

**public:Manager(){**

**perfBonus=16100;**

**}**

**public:Manager(int id,string name,int basicSalary,int deptId,int perfBonus):Employee(id,name,basicSalary,deptId){**

**this->perfBonus=perfBonus;**

**}**

**//function**

**public:double computeNetSalary(){**

**double netSalary=basicSalary+perfBonus;**

**return netSalary;**

**}**

**};**

**class Worker:public Employee{**

**private:**

**int hoursWorked;**

**double hourlyRate;**

**//constructor**

**public:Worker(){**

**hoursWorked=8;**

**hourlyRate=50;**

**}**

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

**this->hoursWorked=hoursWorked;**

**this->hourlyRate=hourlyRate;**

**}**

**//getter**

**public:double getHourlyRate(){**

**return hourlyRate;**

**}**

**//function**

**public:double computeNetSalary(){**

**double netSalary=basicSalary+(hourlyRate\*hoursWorked);**

**return netSalary;**

**}**

**};**

**int main(){**

**Employee \*arr[10];**

**int choice,id,basicSalary,deptId,perfBonus,hoursWorked,hourlyRate,n,i=0;**

**string name;**

**cout<<"1:to hire manager 2:to hire worker 3:display all employee 4:exit"<<endl;**

**do{**

**cout<<"enter choice"<<endl;**

**cin>>choice;**

**switch(choice){**

**case 1:{**

**cout<<"enter id: name: salary: deptId: perfBonus: ";**

**cin>>id>>name>>basicSalary>>deptId>>perfBonus;**

**arr[i]=new Manager(id,name,basicSalary,deptId,perfBonus);**

**i++;**

**break;**

**}**

**case 2:**

**{**

**cout<<"enter id: name: salary: deptId: hoursWorked: hourlyRate: ";**

**cin>>id>>name>>basicSalary>>deptId>>hoursWorked>>hourlyRate;**

**arr[i]=new Worker(id,name,basicSalary,deptId,hoursWorked,hourlyRate);**

**i++;**

**break;**

**}**

**case 3:{**

**for(int j=0;j<i;j++){**

**cout<<arr[j]->computeNetSalary();**

**}**

**break;**

**}**

**case 4:**

**{**

**for(int j=0;j<i;j++){**

**delete arr[j];**

**}**

**cout<<"exit";**

**break;**

**}**

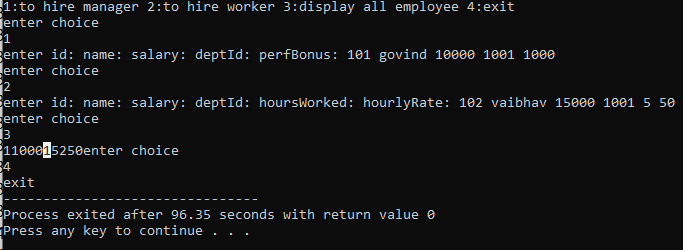
**default:cout<<"invalid";**

**break;**

**}**

**}while(choice!=4);**

**}**

****

**----------------------------------------------------**

**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 BankAccount{**

**private:int acctNo;**

**string name;**

**double balance;**

**//constructor**

**public:BankAccount(){**

**acctNo=101;**

**name="govind";**

**balance=50000;**

**}**

**public:BankAccount(int acctNo,string name,double balance){**

**this->acctNo=acctNo;**

**this->name=name;**

**this->balance=balance;**

**}**

**//function**

**public:void withdraw(int amount){**

**if(amount>balance){**

**cout<<"don't have enough money"<<endl;**

**}else{**

**balance=balance-amount;**

**}**

**}**

**public:void deposit(int amount){**

**balance=balance+amount;**

**}**

**public:void display(){**

**cout<<acctNo<<" "<<name<<" "<<balance<<endl;**

**}**

**};**

**int main(){**

**BankAccount b1(101,"abhi",10000);**

**b1.display();**

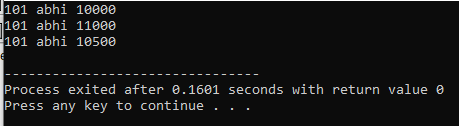
**b1.deposit(1000);**

**b1.display();**

**b1.withdraw(500);**

**b1.display();**

**}**

****

**--------------------------------------------------------------------------**

**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>**

**#include<math.h>**

**using namespace std;**

**class Shape**

**{**

**public:**

**virtual void Area(int a, int b =0) = 0;**

**};**

**class Rectange : public Shape**

**{**

**public:**

**void Area(int a,int b){**

**double area = a \* b;**

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

**}**

**};**

**class Circle: public Shape**

**{**

**public:**

**void Area(int a,int b = 0){**

**int x = pow(a,2);**

**double area = 3.142 \* x;**

**cout<<"Area of Circle is: "<<area<<endl;**

**}**

**};**

**class Square: public Shape**

**{**

**public:**

**void Area(int a, int b = 0){**

**double area = pow(a,2);**

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

**}**

**};**

**int main(){**

**Rectange rc;**

**rc.Area(3,2);**

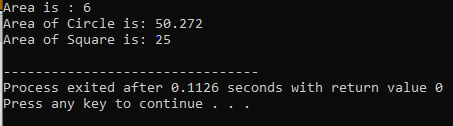
**Circle cc;**

**cc.Area(4);**

**Square sq;**

**sq.Area(5);**

**}**

****

**Day:7**

**1:Create class for Handling Exception for Student**

**Create StudentException class;**

**Create Student class with data members (rollno,name,age)**

**at the time of student creation check age if age<18 then throw exception**

**#include<iostream>**

**using namespace std;**

**class StudentException:public exception{**

**public: void ageException(){**

**cout<<"invalid age"<<endl;**

**cout<<"age should be greater than 18";**

**}**

**};**

**class Student{**

**StudentException \*se1=new StudentException;**

**private:int rollno,age;**

**string name;**

**public: Student(){**

**rollno=101;**

**name="vaibhav";**

**age=19;**

**}**

**Student(int rollno,string name,int age){**

**this->rollno=rollno;**

**this->name=name;**

**this->age=age;**

**if(age<18){**

**throw se1;**

**}**

**}**

**};**

**int main(){**

**int rollno,age;**

**string name;**

**cout<<"enter rollno,name,age";**

**cin>>rollno>>name>>age;**

**try{**

**Student s1(rollno,name,age);**

**}**

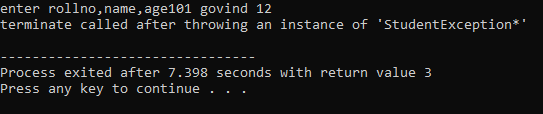
**catch(StudentException &se1){**

**se1.ageException();**

**}catch(...){**

**}**

**}**

****

**2:Create Function Template and class template and use it**

**#include<iostream>**

**using namespace std;**

**template <class T> class SwapClass{**

**public:SwapClass(){**

**}**

**public:void swapT(T &a,T &b){**

**cout<<"before swap value of a and b "<<a<<" "<<b<<endl;**

**T temp=a;**

**a=b;**

**b=temp;**

**cout<<"after swap value of a and b "<<a<<" "<<b<<endl;**

**}**

**};**

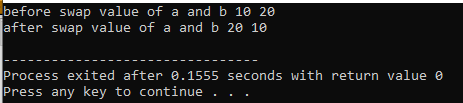
**int main(){**

**SwapClass<int> sc;**

**int a=10,b=20;**

**sc.swapT(a,b);**

**};**

****

**3:Create abstract class Fruit with abstract function taste**

**Create derived classes for Fruit:Apple,Mango,Orange**

**Implement abstract method in derived class ,add specific methods like juice(),jam(),pulp() in respective derived class(as discussed in lect);**

**create taster program.create fruit basket and invoke method**

**Use:dynamic\_cast,typeid**

#include<iostream>

#include<typeinfo>

using namespace std;

class Fruit{

public:virtual void taste()=0;

};

class Apple:public Fruit{

public:void taste(){

cout<<"apple taste"<<endl;

}

void jam(){

cout<<"jam"<<endl;

}

};

class Orange:public Fruit{

public:void taste(){

cout<<"orange taste"<<endl;

}

void juice(){

cout<<"juice"<<endl;

}

};

class Mango:public Fruit{

public:void taste(){

cout<<"mango taste"<<endl;

}

void pulp(){

cout<<"pulp"<<endl;

}

};

int main(){

Fruit \*arr[10];

int choice,i=0;

do{

cout<<"enter choice"<<endl;

cout<<"1:add apple"<<endl;

cout<<"2:add mango"<<endl;

cout<<"3:add orange"<<endl;

cout<<"4:display output of method"<<endl;

cout<<"5:exit"<<endl;

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

cin>>choice;

switch (choice){

case 1:{

arr[i]=new Apple();

cout<<"Apple added"<<endl;

i++;

break;

}

case 2:{

arr[i]=new Mango();

cout<<"Mango added"<<endl;

i++;

break;

}

case 3:{

arr[i]=new Orange();

cout<<"Orange added"<<endl;

i++;

break;

}

case 4:{

for(int j=0;j<i;j++){

if(typeid(\*arr[j])==typeid(Mango)){

Mango \*m1=dynamic\_cast<Mango\*>(arr[j]);

m1->pulp();

m1->taste();

}

else if(typeid(\*arr[j])==typeid(Apple)){

Apple \*a1=dynamic\_cast<Apple\*>(arr[j]);

a1->jam();

a1->taste();

}

else if(typeid(\*arr[j])==typeid(Orange)){

Orange \*o1=dynamic\_cast<Orange\*>(arr[j]);

o1->juice();

o1->taste();

}

}

cout<<endl<<endl;

break;

}

case 5:{

cout<<"exit"<<endl;

break;

}

default:{

cout<<"invalid input"<<endl;

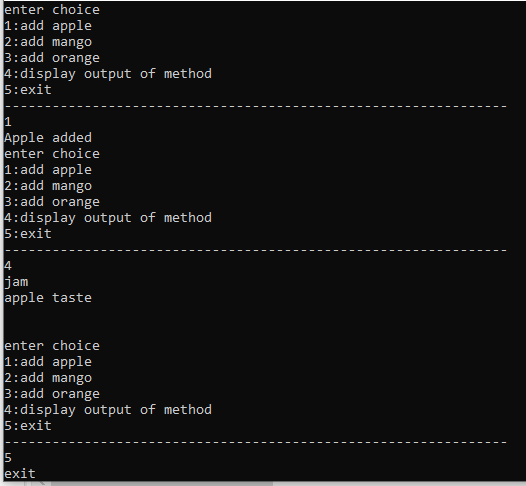
break;

}

}

}while(choice!=5);

}



**Day:8**

**1 Write a program to store characters 'A' to 'Z' in the file .**

**Display the contents of file.**

**#include<iostream>**

**#include<fstream>**

**using namespace std;**

**int main(){**

**char och,ch='A';**

**int i;**

**ofstream o1("alphabate.txt",ios::app);**

**for(i=0;i<25;i++){**

**o1<<char(ch+i);**

**}**

**o1.close();**

**ifstream i1("alphabate.txt",ios::app);**

**while(i1){**

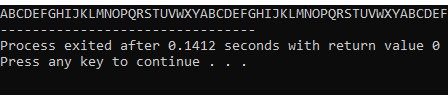
**i1>>och;**

**cout<<och;**

**}**

**i1.close();**

**}**

****

**2. Create class cEmployee with data members as empno,name and salary.**

**Accept values from user. Store it in file.**

**Display the contents from file.**

**Program should be able to store information of multiple**

**Employees.**

**#include<iostream>**

**#include<fstream>**

**using namespace std;**

**class cEmployee{**

**private:int empno,salary;**

**string name;**

**public:cEmployee(){**

**empno=101;**

**name="govind";**

**salary=100000;**

**}**

**public:cEmployee(int empno,string name,int salary){**

**this->empno=empno;**

**this->name=name;**

**this->salary=salary;**

**}**

**public:void display(){**

**cout<<empno<<" "<<name<<" "<<salary<<" "<<endl;**

**}**

**};**

**int main(){**

**cEmployee emp1(102,"ram",200000);**

**ofstream of1("Employee.txt",ios::binary);**

**of1.write((char\*)&emp1,sizeof(emp1));**

**of1.close();**

**ifstream if1("Employee.txt",ios::binary);**

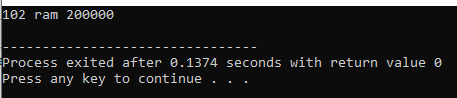
**if1.seekg(0,ios::beg);**

**if1.read((char\*)&emp1,sizeof(emp1));**

**emp1.display();**

**if1.close();**

**}**

****

**3:Write a program to copy the contents of one file and write it into another file and print it.**

**#include<iostream>**

**using namespace std;**

**class Student{**

**public:int id;**

**string name;**

**public:Student(){**

**id=101;**

**name="govind";**

**}**

**Student(int id,string name ){**

**this->id=id;**

**this->name=name;**

**}**

**Student(Student &obj){**

**this->id=obj.id;**

**this->name=obj.name;**

**}**

**public:void display(){**

**cout<<id<<name;**

**}**

**};**

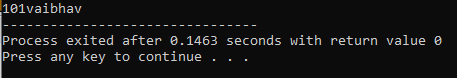
**int main(){**

**Student s1(101,"vaibhav");**

**Student s2(s1);**

**s2.display();**

**}**

****

**4:Create a C++ program that does the following:**

**Initializes an empty vector of integers.**

**Asks the user to enter a series of integers and adds them to the vector until the user enters a specific sentinel value (e.g., -1).**

**Prints the elements of the vector.**

**Calculates and prints the sum and average of the values in the vector.**

**#include<iostream>**

**#include<vector>**

**#include<Iterator>**

**using namespace std;**

**void calculate(vector<int> &v1){**

**int sum=0;**

**double avg;**

**for(int i=0;i<v1.size();i++){**

**sum=sum+v1[i];**

**}**

**cout<<"sum: "<<sum<<endl;**

**avg=double(sum)/v1.size();**

**cout<<"avg: "<<avg<<endl;**

**}**

**int main(){**

**vector<int> v1;**

**int no;**

**do{**

**cout<<"enter no which is greater than zero: ";**

**cin>>no;**

**v1.push\_back(no);**

**}while(no>0);**

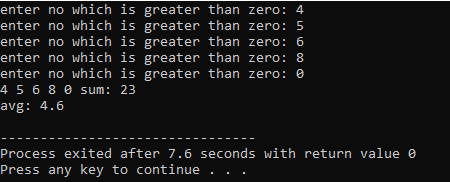
**for(int i=0;i<v1.size();i++){**

**cout<<v1[i]<<" ";**

**}**

**calculate(v1);**

**}**

****

**5:Defines a Student class with attributes like name, age, and grade.**

**Initializes a vector of Student objects.**

**Allows the user to add new students to the vector with their name, age, and grade.**

**Prints the list of students.**

**#include<iostream>**

**#include<vector>**

**using namespace std;**

**class Student{**

**private:string name;**

**int age;**

**char ch;**

**public:Student(){**

**name="govind";**

**age=18;**

**ch='A';**

**}**

**public:Student(string name,int age,char ch){**

**this->name=name;**

**this->age=age;**

**this->ch=ch;**

**}**

**public:void display(){**

**cout<<name<<" "<<age<<" "<<ch<<endl;**

**}**

**};**

**int main(){**

**vector<Student> v1;**

**Student s1("ram",10,'B');**

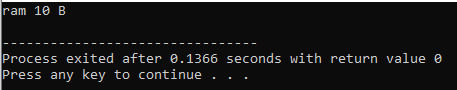
**v1.push\_back(s1);**

**for(int i=0;i<v1.size();i++){**

**v1[i].display();**

**}**

**}**

****