

***SUBMITTED TO:- SUBMITTED BY:-***

Dr. Sanjay katyal Santosh sharma Computer science MCA (Ist) Semester

Department Roll no. 401



***SUBMITTED TO:- SUBMITTED BY:-***

Dr. Ashok sssSantosh sharma Computer science MCA (Ist) Semester

Department Roll no. 401



***SUBMITTED TO:- SUBMITTED BY:-***

Dr. Sanjay katyal Santosh sharma Computer science MCA (Ist) Semester

Department Roll no. 401

***Raising a number n to a power p is the same as multiplying n by itself p times. Write a function called power ( ) that takes a double value for n and an int value for p, and returns the result as double value. Use a default argument of 2 for p, so that if this argument is omitted, the number will be squared. Write a main ( ) function that gets values from the user to test this function.***

***PROGRAM:-***

***#include<iostream.h>***

***#include<conio.h>***

***double power(double n,int p=2);***

***int main()***

***{***

***doublen,r;***

***int p;***

***char c;***

***clrscr();***

***cout<<"enter the number:\n";***

***cin>>n;***

***do***

***{***

***cout<<"do you want to enter power(y/n)?:\n";***

***cin>>c;***

***if(c=='y')***

***{***

***cout<<"enter the power to be raised:\n";***

***cin>>p;***

***r=power(n,p);***

***}***

***else***

***{***

***if(c=='n')***

***{p=2;***

***r=power(n);***

***}***

***else***

***cout<<"invalid choice\n";***

***}***

***}while(c!='y'&&c!='n');***

***cout<<n<<"^"<<p<<"("<<n<<" raised to the power "<<p<<")="<<r;***

***getch();***

***return 0;***

***}***

***double power(double n,int p)***

***{***

***double r=1;***

***int i;***

***if(p<0)***

***r=1/power(n,-p);***

***else***

***for(i=1;i<=p;i++)***

***r=r\*n;***

***return(r);***

***}***

**OUTPUT:-**

**Write a program to print a triangles like the followings using function overloading: -**

1. **1**

**1 2**

**1 2 3**

**1 2 3 4**

**1 2 3 4 5**

1. **1**

**2 3**

**4 5 6**

**7 8 9 10**

**11 12 13 14**

1. **1**

**1 1 1**

**1 1 1 1 1**

**1 1 1 1 1 1 1**

**1 1 1 1 1 1 1 1 1**

1. **1**

**1 2 1**

**1 2 3 2 1**

**1 2 3 4 3 2 1**

**1 2 3 4 5 4 3 2 1**

**PROGRAM:-**

**(a)**

**#include<iostream.h>**

**int main()**

**{**

**int rows;**

**cout<< "Enter number of rows: ";**

**cin>> rows;**

**for(int i = 1; i <= rows; ++i)**

**{**

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

**{**

**cout<< j << " ";**

**}**

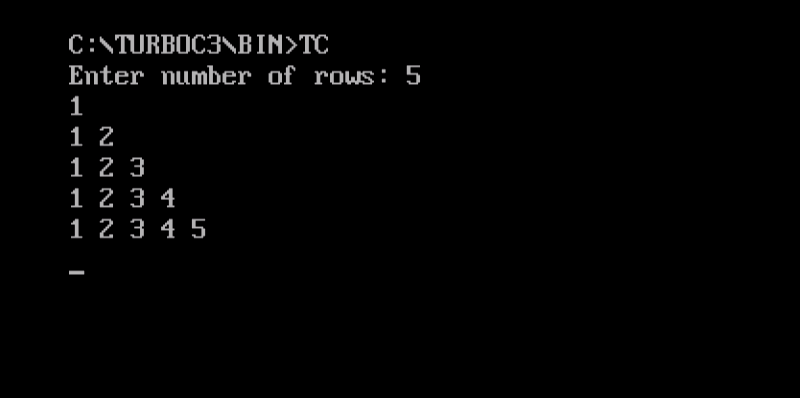
**cout<< "\n";**

**}**

**return 0;**

**}**

**Output:-**

****

**(b).**

**#include<iostream.h>**

**#include<conio.h>**

**void main()**

**{**

**inti,j;**

**int n=1;**

**for(i=1;i<=5;i++)**

**{**

**for(j=1;j<=i;j++)**

**{**

**cout<<n;**

**n++;**

**}**

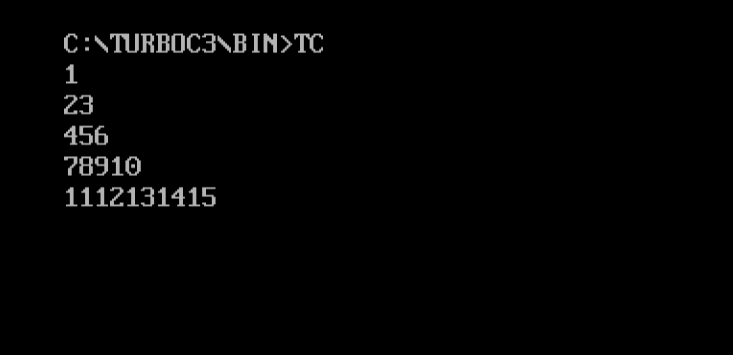
**cout<<endl;**

**}**

**getch();**

**}**

**Output:-**

****

**(c).**

**#include<iostream.h>**

**#include<conio.h>**

**void main()**

**{**

**intmin\_stars=1;**

**intp\_height=5;**

**intp\_space=p\_height-1;**

**inti,j,k;**

**for(i=0;i<p\_height;i++)**

**{**

**for(j=p\_space;j>i;j--)**

**{**

**cout<<"";**

**}**

**for(k=0;k<min\_stars;k++)**

**{**

**cout<<"1";**

**}**

**min\_stars+=2;**

**cout<<endl;**

**}**

**getch();**

**}**

**Output:-**

****

**(d)**

**#include<iostream.h>**

**#include<conio.h>**

**#include<math.h>**

**void main()**

**{**

**int z=1;**

**for(int i=1;i<=5;i++)**

**{**

**for(int j=4;j>=i;j--)**

**{**

**cout<<"";**

**}**

**for(int k=i-1;k>=-(i-1);k--)**

**{**

**cout<<i-abs(k);**

**}**

**z+=2;**

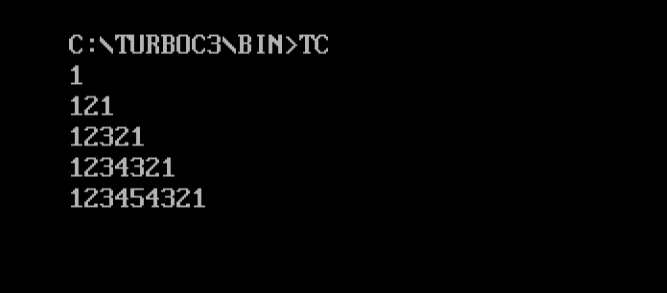
**cout<<endl;**

**}**

**getch();**

**}**

**OUTPUT:-**

****

1. **A point on the two dimensional plane can be represented by two numbers: an X coordinate and a Y coordinate. For example, (4,5) represents a point 4 units to the right of the origin along the X axis and 5 units up the Y axis. The sum of two points can be defined as a new point whose X coordinate is the sum of the X coordinates of the points and whose Y coordinate is the sum of their Y coordinates. Write a program that uses a structure called point to model a point. Define three points, and have the user input values to two of them. Than set the third point equal to the sum of the other two, and display the value of the new point. Interaction with the program might look like this:**

**Enter coordinates for P1: 3 4**

**Enter coordinates for P2: 5 7**

**Coordinates of P1 + P2 are : 8, 11**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**#define N 2**

**struct point**

**{**

**int x;**

**int y;**

**}p[N],pt={0,0};**

**int main()**

**{**

**int i;**

**clrscr();**

**for(i=0;i<=N-1;i++)**

**{**

**cout<<"enter coordinates x"<<i+1<<" & y"<<i+1<<":";**

**cin>>p[i].x>>p[i].y;**

**}**

**for(i=0;i<=N-1;i++)**

**{**

**pt.x=pt.x+p[i].x;**

**pt.y=pt.y+p[i].y;**

**}**

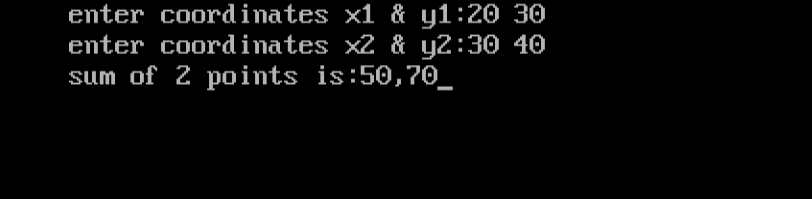
**cout<<"sum of "<<N<<" points is:"<<pt.x<<","<<pt.y;**

**getch();**

**return 0;**

**}**

**OUTPUT:-**

****

1. **Write a program to find out the gross amount from the given basic pay.**

**Gross = Basic + DA + HRA**

**DA & HRA can be calculated as follows: -**

**If Basic Pay is greater than or equal to 8000 DA is 20% of Basic Pay & HRA is 25% of Basic Pay, otherwise DA is 15% of Basic Pay & HRA is 20% of Basic Pay.**

**Using three functions input, calculate and display.**

**#include<iostream>**

**#include<stdlib.h>**

**using namespace std;**

**class salary**

**{**

**float Basic,HRA,DA,Gross;**

**public:**

**void input()**

**{**

**cout<<"Enter the BASIC salary of employee"<<endl;**

**cin>>Basic;**

**}**

**void calculate()**

**{**

**if(Basic>=8000)**

**{ DA=Basic\*(float)20/100;**

**HRA=Basic\*(float)25/100;**

**Gross=Basic+DA+HRA;**

**}**

**else**

**{**

**DA=Basic\*(float)15/100;**

**HRA=Basic\*(float)20/100;**

**Gross=Basic+DA+HRA;**

**}**

**}**

**void display()**

**{**

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

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

**cout<<"Gross salary is : "<<Gross<<endl;**

**} };**

**int main()**

**{ salary s;**

**s.input();**

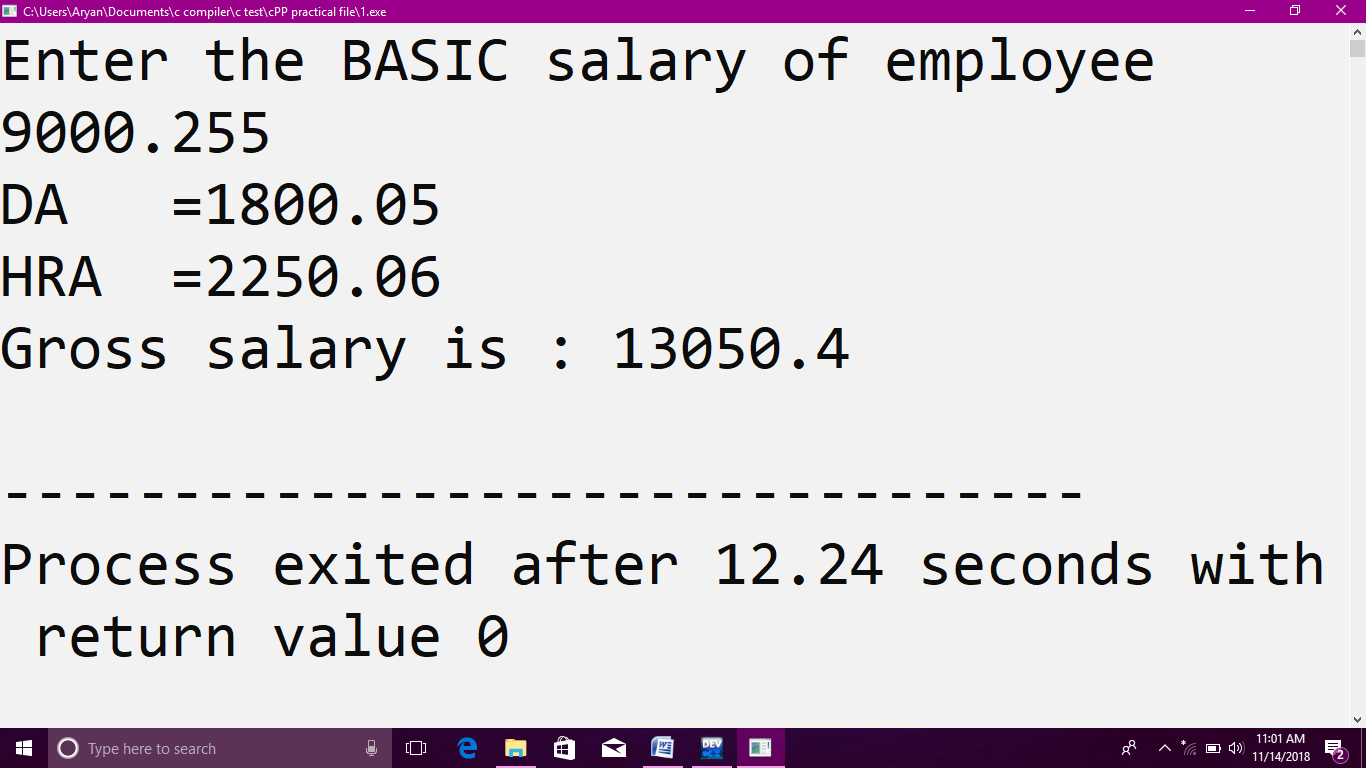
**s.calculate();**

**s.display();**

**return 0;**

**}**

**OUTPUT:**

****

1. **Write a program for Multiplication of the two matrixes and store it in another matrix. Display the new matrix using three functions one for input, second for multiplication with array as argument and display for output.**

**PROGRAM:-**

**#include <iostream.h>**

**int main()**

**{**

**int a[10][10], b[10][10], mult[10][10], r1, c1, r2, c2, i, j, k;**

**cout<< "Enter rows and columns for first matrix: ";**

**cin>> r1 >> c1;**

**cout<< "Enter rows and columns for second matrix: ";**

**cin>> r2 >> c2;**

**while (c1!=r2)**

**{**

**cout<< "Error! column of first matrix not equal to row of second.";**

**cout<< "Enter rows and columns for first matrix: ";**

**cin>> r1 >> c1;**

**cout<< "Enter rows and columns for second matrix: ";**

**cin>> r2 >> c2;**

**}**

**cout<<endl<< "Enter elements of matrix 1:" <<endl;**

**for(i = 0; i < r1; ++i)**

**for(j = 0; j < c1; ++j)**

**{**

**cout<< "Enter element a" << i + 1 << j + 1 << " : ";**

**cin>> a[i][j];**

**}**

**cout<<endl<< "Enter elements of matrix 2:" <<endl;**

**for(i = 0; i < r2; ++i)**

**for(j = 0; j < c2; ++j)**

**{**

**cout<< "Enter element b" << i + 1 << j + 1 << " : ";**

**cin>> b[i][j];**

**}**

**for(i = 0; i < r1; ++i)**

**for(j = 0; j < c2; ++j)**

**{**

**mult[i][j]=0;**

**}**

**for(i = 0; i < r1; ++i)**

**for(j = 0; j < c2; ++j)**

**for(k = 0; k < c1; ++k)**

**{**

**mult[i][j] += a[i][k] \* b[k][j];**

**}**

**cout<<endl<< "Output Matrix: " <<endl;**

**for(i = 0; i < r1; ++i)**

**for(j = 0; j < c2; ++j)**

**{**

**cout<< " " <<mult[i][j];**

**if(j == c2-1)**

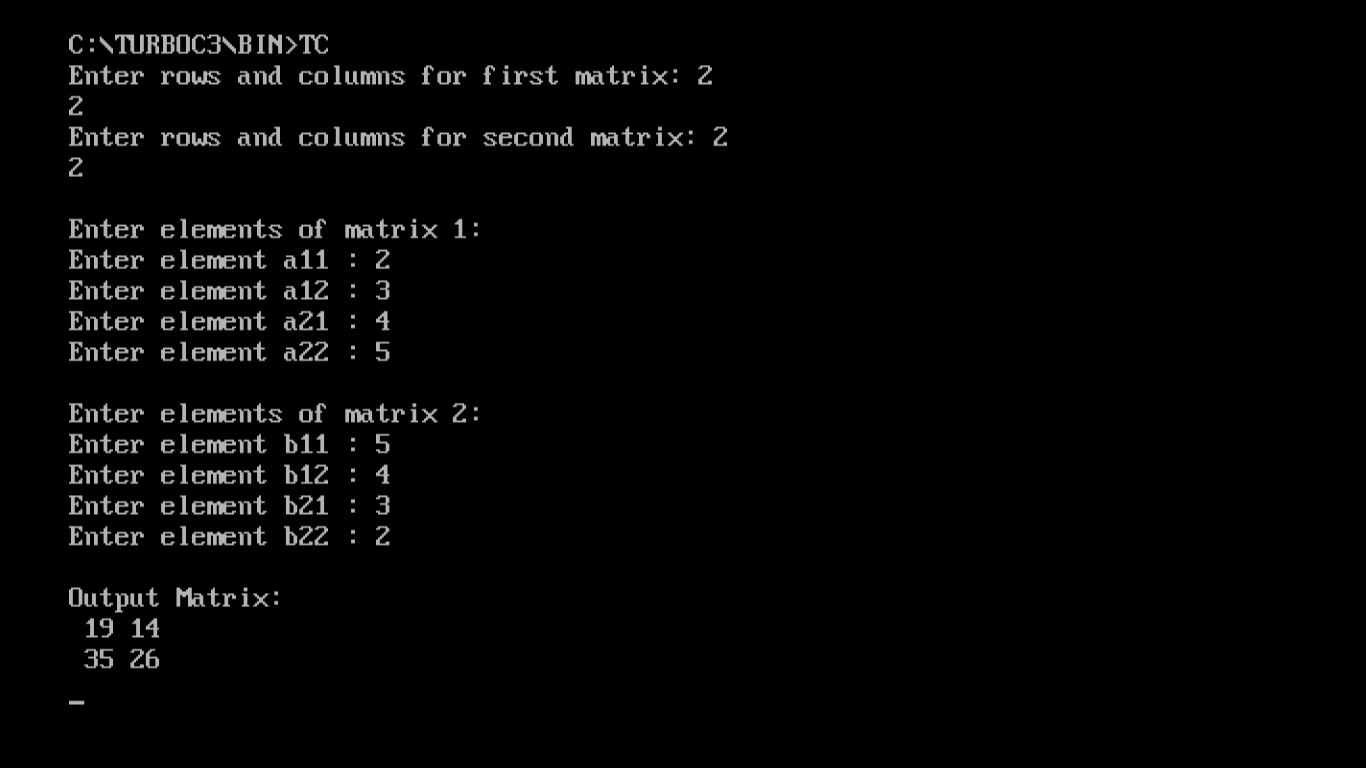
**cout<<endl;**

**}**

**return 0;**

**}**

**OUTPUT:-**

****

1. **Create the equivalent of a four function calculator. The program should request the user to enter a number, an operator, and another number. It should then carry out the specified arithmetical operation: adding, subtracting, multiplying, or dividing the two numbers. (It should use a switch statement to select the operation). Finally it should display the result. When it finishes the calculation, the program should ask if the user wants to do another calculation. The response can be ‘Y’ or ‘N’. Some sample interaction with the program might look like this.**

**Enter first number, operator, second number: 10/ 3**

**Answer = 3.333333**

**Do another (Y/ N)? Y**

**Enter first number, operator, second number 12 + 100**

**Answer = 112**

**Do another (Y/ N)? N**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**int sum(int,int);**

**int sub(int,int);**

**intmul(int,int);**

**int div(int,int);**

**int main()**

**{**

**inta,c;**

**charb,ch;**

**clrscr();**

**do**

**{**

**cout<<"enter a number ,operator and second number"<<endl;**

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

**cout<<"result=";**

**switch(b)**

**{**

**case'+':**

**cout<<sum(a,c)<<endl;**

**break;**

**case'-':**

**cout<<sub(a,c)<<endl;**

**break;**

**case'\*':**

**cout<<mul(a,c)<<endl;**

**break;**

**case'/':**

**cout<<div(a,c)<<endl;**

**break;**

**default:**

**cout<<"wrong choice";**

**}**

**cout<<"do you want to continue (y/n) "<<endl;**

**cin>>ch;**

**}**

**while (ch=='y'||ch=='Y');**

**getch();**

**}**

**int sum(inta,int c)**

**{**

**return(a+c);**

**}**

**int sub(inta,int c)**

**{**

**return(a-c);**

**}**

**intmul(inta,int c)**

**{**

**return(a\*c);**

**}**

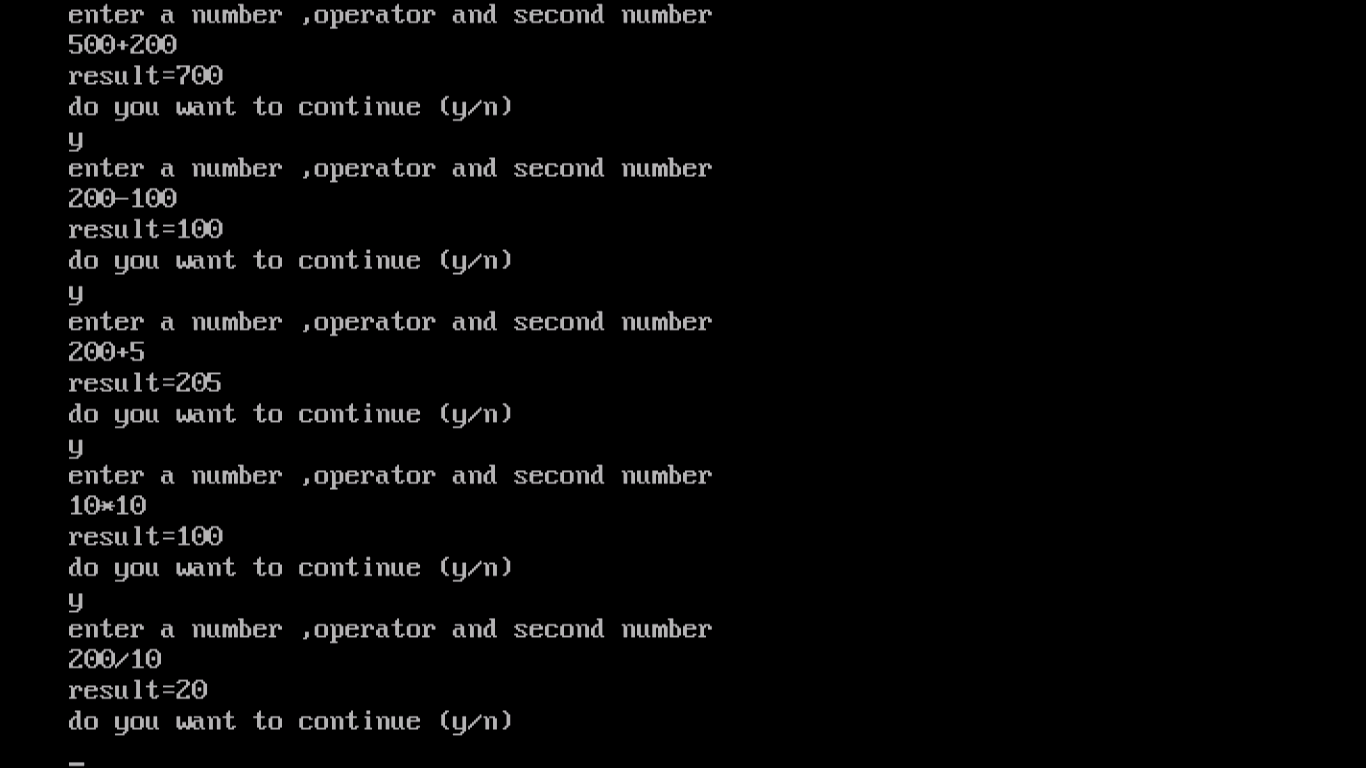
**int div(inta,int c)**

**{**

**return(a/c);**

**}**

**OUTPUT:-**

****

1. **A phone number, such as (212) 767-8900, can be thought of as having three parts: the area code (212), the exchange (767) and the number (8900). Write a program that uses a structure to store these three parts of a phone number separately. Call the structure phone. Create two structure variables of type phone. Initialize one, and have the user input a number for the other one. Then display both numbers. The interchange might look like this:**

**Enter your area code, exchange, and number: 415 555 1212**

**My number is (212) 767-8900**

**Your number is (415) 555-1212**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**struct phone**

**{**

**char area[10];**

**char exchange[10];**

**char number[10];**

**};**

**int main()**

**{**

**phone ph1={"212","767","8900"};**

**phone ph2;**

**clrscr();**

**cout<<"\nenter your area code,exchange and number:\n";**

**cin>>ph2.area>>ph2.exchange>>ph2.number;**

**cout<<"\nmy number is ("<<ph1.area<<")"<<ph1.exchange<<"-"<<ph1.number;**

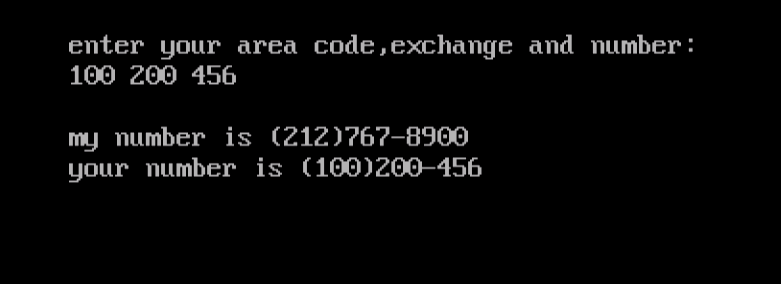
**cout<<"\nyour number is ("<<ph2.area<<")"<<ph2.exchange<<"-"<<ph2.number;**

**getch();**

**return 0;**

**}**

**OUTPUT:-**

[****](http://1.bp.blogspot.com/-qojnwJhOrmY/T2SxpWBPhmI/AAAAAAAAACQ/0Ky0uuqfmOc/s1600/3.jpg)

1. **Create two classes DM and DB which store the value of distances. DM stores distances in metres and centimeters and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB. Use a friend function to carry out the addition operation. The object that stores the results May be a DM object or DB object, depending on the units in which the results are required. The display should be in the format of feet and inches or metres and centimetres depending on the object on display.**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**class DB;**

**class DM**

**{**

**intmet,cent;**

**public:**

**void input();**

**voiddisp();**

**friend void sum(DM&,DB);**

**friend void sum(DB&,DM);**

**};**

**void DM::input()**

**{**

**cout<<"Enter distace in metres, centimeters: "<<endl;**

**cin>>met>>cent;**

**}**

**void DM::disp()**

**{**

**cout<<"The distance in metres and centimeters is: "<<met<<"metres and "<<cent<<"centimeters"<<endl;**

**}**

**class DB**

**{**

**intfeet,inch;**

**public:**

**void input();**

**voiddisp();**

**friend void sum(DM&,DB);**

**friend void sum(DB&,DM);**

**};**

**void DB::input()**

**{**

**cout<<"Enter distace in feet, inches: "<<endl;**

**cin>>feet>>inch;**

**}**

**void DB::disp()**

**{**

**cout<<"The distance in feet and inches is: "<<feet<<"feet and "<<inch<<"inches"<<endl;**

**}**

**void sum(DM&dm,DBdb)**

**{**

**float x=db.feet\*30.48;**

**x+=db.inch\*2.54;**

**int y=x;**

**dm.met+=y/100;**

**dm.cent+=y%100;**

**dm.met+=dm.cent/100;**

**dm.cent=dm.cent%100;**

**}**

**void sum(DB&db,DMdm)**

**{**

**float x=dm.met\*39.37;**

**x+=dm.cent\*0.3937;**

**int y=x;**

**db.feet+=y/12;**

**db.inch+=y%12;**

**db.feet+=db.inch/12;**

**db.inch=db.inch%12;**

**}**

**void main()**

**{**

**DM d1;**

**DB d2;**

**int a;**

**clrscr();**

**d1.input();**

**d2.input();**

**cout<<"Enter 1 to get output in metres or enter 2 for feet format: "<<endl;**

**cin>>a;**

**if (a==1)**

**{**

**sum(d1,d2);**

**d1.disp();**

**}**

**else**

**{**

**sum(d2,d1);**

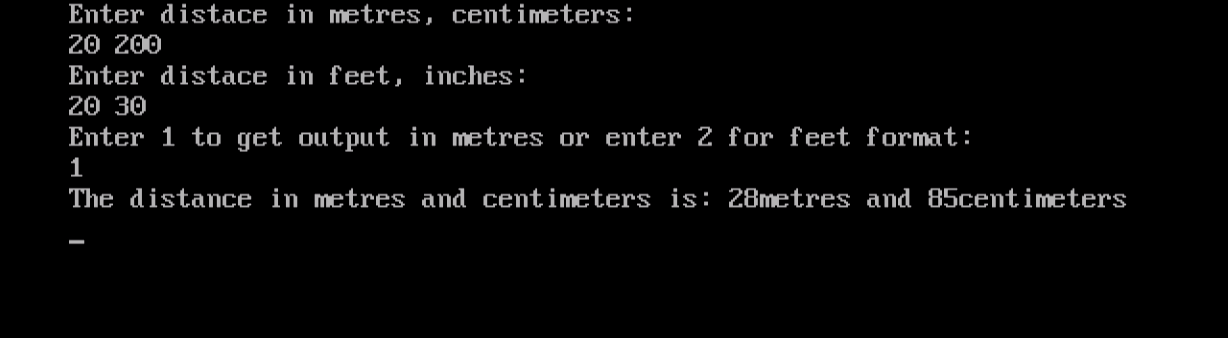
**d2.disp();**

**}**

**getch();**

**}**

**OUTPUT:-**

****

1. **Create a class rational which represents a numerical value by two double values-**

**NUMERATOR &  DENOMINATOR.**

**Include the following public member Functions: constructor with no arguments (default). constructor with two arguments. void reduce( ) that reduces the rational number by eliminating the highest common factor between the numerator and denominator. Overload + operator to add two rational number.**

**Overload >> operator to enable input through cin.**

**Overload << operator to enable output through cout.**

**Write a main ( ) to test all the functions in the class.**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**#include<process.h>**

**class rational**

**{**

**private:**

**intnum;**

**int den;**

**int lcm(inta,int b);**

**inthcf(inta,int b);**

**public:**

**rational()**

**{**

**num=0;**

**den=1;**

**}**

**rational(intn,int d=1)if(d==0)**

**{**

**{**

**cout<<"denominator can't be zero";**

**getch();**

**exit(1);**

**}**

**else**

**{**

**num=n;**

**den=d;**

**}**

**}**

**rational reduce()**

**{**

**rational temp;**

**int h=hcf(num,den);**

**temp.num=num/h;**

**temp.den=den/h;**

**return temp;**

**}**

**rational operator +(rational r)**

**{**

**int l,t1,t2,x1,x2,den1,den2,p;**

**rationalans;**

**if(den<0)**

**den1=-den;**

**else**

**den1=den;**

**if(r.den<0)**

**den2=-r.den;**

**else**

**den2=r.den;**

**p=lcm(den1,den2);**

**if((den<0&&r.den>0)||(den>0&&r.den<0))**

**l=-p;**

**else**

**l=p;**

**ans.den=l;**

**t1=l/den;**

**t2=l/r.den;**

**x1=num\*t1;**

**x2=r.num\*t2;**

**ans.num=x1+x2;**

**returnans;**

**}**

**friendistream& operator >> (istream&s,rational& r);**

**friendostream& operator << (ostream&s,rational& r);**

**};**

**int rational::lcm(inta,int b)**

**{**

**inti,lcm=1;**

**while(!(a==1&&b==1))**

**{**

**i=2;**

**while(!(a%i==0||b%i==0)&&i<(a>b?a:b))**

**i++;**

**lcm\*=i;**

**if(a%i==0)**

**a/=i;**

**if(b%i==0)**

**b/=i;**

**}**

**return lcm;**

**}**

**int rational::hcf(inta,int b)**

**{**

**int r=a%b;**

**while(r)**

**{**

**a=b;**

**b=r;**

**r=a%b;**

**}**

**return b;**

**}**

**istream& operator >>(istream&s,rational& r)**

**{**

**inta,b;**

**char c;**

**s>>a>>c>>b;**

**if(c!='/')**

**{**

**cout<<"use of invalid notation";**

**getch();**

**exit(0);**

**}**

**if(b==0)**

**{**

**cout<<"denominator can't be zero.";**

**getch();**

**exit(1);**

**}**

**r.num=a;**

**r.den=b;**

**return s;**

**}**

**ostream& operator <<(ostream&s,rational& r)**

**{**

**if(r.den==1)**

**s<<r.num;**

**else**

**{**

**if(r.den==-1)**

**s<<-r.num;**

**else**

**s<<r.num<<'/'<<r.den;**

**}**

**return s;**

**}**

**int main()**

**{**

**clrscr();**

**rational r1,r2,r3;**

**cout<<"enter r1:";**

**cin>>r1;**

**cout<<"enter r2:";**

**cin>>r2;**

**r3=r1+r2;**

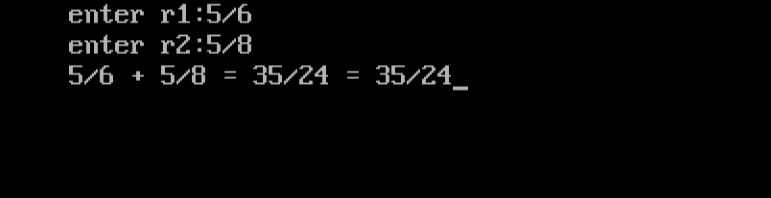
**cout<<r1<<" + "<<r2<<" = "<<r3<<" = "<<r3.reduce();**

**getch();**

**return 0;**

**}**

**OUTPUT:-**

****

1. **Consider the following class definition**

**class father {**

**protected :int age;**

**public;**

**father (int x) {age = x;}**

**virtual void iam ( )**

**{ cout<< I AM THE FATHER, my age is : << age<< end1:}**

**};**

**Derive the two classes son and daughter from the above class and for each, define iam ( ) to write our similar but appropriate messages. You should also define suitable constructors for these classes. Now, write a main ( ) that creates objects of the three classes and then calls iam ( ) for them. Declare pointer to father. Successively, assign addresses of objects of the two derived classes to this pointer and in each case, call iam ( ) through the pointer to demonstrate polymorphism in action.**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**class father**

**{**

**protected:unsignedint age;**

**public:**

**father()**

**{**

**age=60;**

**}**

**father(int x)**

**{**

**age=x;**

**}**

**virtual void iam()**

**{**

**cout<<"I AM THE FATHER,my age is:"<<age<<endl;**

**}**

**};**

**classson:public father**

**{**

**public:**

**son()**

**{**

**age=30;**

**}**

**son(int x)**

**{**

**age=x;**

**}**

**voidiam()**

**{**

**cout<<"I AM THE SON,my age is:"<<age<<endl;**

**}**

**};**

**classdaughter:public father**

**{**

**public:**

**daughter()**

**{**

**age=24;**

**}**

**daughter(int x)**

**{**

**age=x;**

**}**

**voidiam()**

**{**

**cout<<"I AM THE DAUGHTER,my age is:"<<age<<endl;**

**}**

**};**

**int main()**

**{**

**father f(50),\*ptrf;**

**son s(23);**

**daughter d(16);**

**clrscr();**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CALL BY FATHER OBJECT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**f.iam();**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CALL BY SON OBJECT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**s.iam();**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CALL BY DAUGHTER OBJECT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**d.iam();**

**ptrf=&s;**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CALL BY POINTER TO FATHER WITH ADDRESS OF SON OBJECT\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**ptrf->iam();//(\*ptrf).iam();**

**cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*CALL BY POINTER TO FATHER WITH ADDRESS OF DAUGHTER OBJECT\*\*\*\*\*\*\*\*\*\*\*\n";**

**ptrf=&d;**

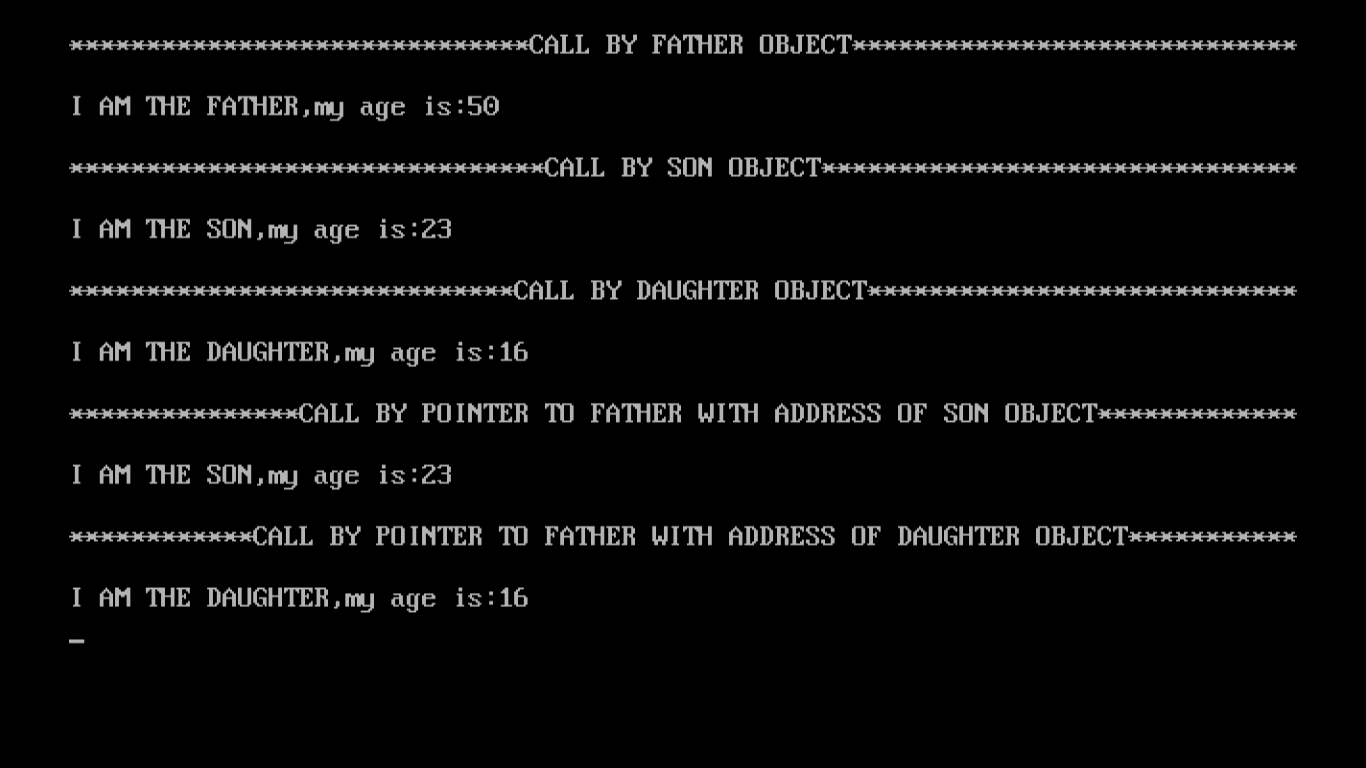
**ptrf->iam();//(\*ptrf).iam();**

**getch();**

**return 0;**

**}**

**OUTPUT:-**

****

1. **Write a program that creates a binary file by reading the data for the students from the terminal. The data of each student consist of roll no., name ( a string of 30 or lesser no. of characters) and marks**

**.**

**PROGRAM:-**

**#include<iostream.h>**

**#include<fstream.h>**

**#define m 2**

**class student**

**{**

**introllno, marks;**

**char nm[30];**

**public:**

**voidgetdata()**

**{**

**cout<<"\n Enter roll no. : ";**

**cin>>rollno;**

**charch=cin.get();**

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

**cin.getline(nm,30);**

**cout<<"\n Enter marks : ";**

**cin>>marks;**

**}**

**};**

**int main()**

**{**

**studentstu[m];**

**fstreamfio;**

**fio.open("Student.txt",ios::binary|ios::out);**

**for(int count=0;count<m;count++)**

**{**

**stu[count].getdata();**

**fio.write((char \*)&stu[count],sizeof(stu[count]));**

**}**

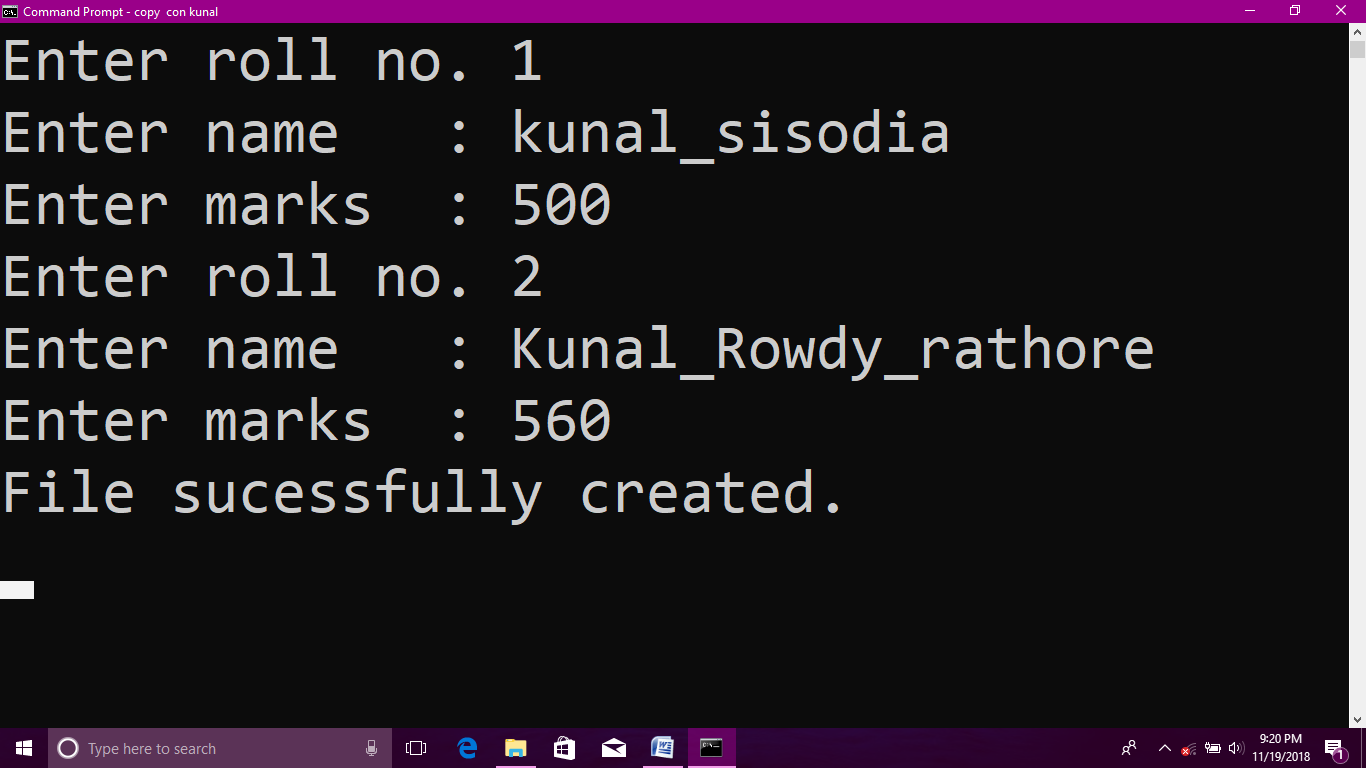
**fio.close();**

**cout<<”File successfully created”<<endl;**

**return 0;**

**}**

**Output:**

****

1. **A hospital wants to create a database regarding its indoor patients. The information to store include:**

**a)      Name of the patient**

**b)      Date of admission.**

**c)       Disease.**

**d)      Date of discharge.**

**Create a structure to store the date (year, month and date as its members). Create a base class to store the above information. The member function should include functions to enter information and display a list of all the patients in the database. Create a derived class to store the age f the patients. List the information about all to store the age of patients. List the information about all the pediatric patients (less than twelve years in age).**

**PROGRAM:-**

**#include<iostream.h>**

**#include<stdio.h>**

**#include<conio.h>**

**strucdate**

**{**

**intyear;**

**intmonth;**

**int date;**

**};**

**class patient**

**{**

**protected:**

**char name[20];**

**dateadm;**

**char disease[20];**

**date dis;**

**public:**

**voidgetdata();**

**voidshowdata();**

**};**

**classnew\_patient:public patient**

**{**

**int age;**

**public:**

**voidgetage();**

**voidshowinfo();**

**};**

**void patient::getdata()**

**{**

**cout<<"\nEnter the name of the patient ::";**

**gets(name);**

**cout<<"\nEnter the date of addmission of the patient ::";**

**cin>>adm.date;**

**cout<<"\nEnter the month of addmission of the patient ::";**

**cin>>adm.month;**

**cout<<"\nEnter the year of addmission of the patient ::";**

**cin>>adm.year;**

**cout<<"\nEnter the disease of the patient ::";**

**gets(disease);**

**cout<<"\nEnter the date of discharge of the patient ::";**

**cin>>dis.date;**

**cout<<"\nEnter the month of discharge of the patient ::";**

**cin>>dis.month;**

**cout<<"\nEnter the year of discharge of the patient ::";**

**cin>>dis.year;**

**}**

**void patient::showdata()**

**{**

**cout<<"\nThe name of the patient is:="<<name;**

**cout<<"\nThe date of addmission of the patient is:="<<adm.date;**

**cout<<"-"<<adm.month;**

**cout<<"-"<<adm.year;**

**cout<<"\nThe disease of the patient is:="<<disease;**

**cout<<"\nThe date of discharge of the patient ::"<<dis.date;**

**cout<<"-"<<dis.month;**

**cout<<"-"<<dis.year;**

**}**

**voidnew\_patient::getage()**

**{**

**cout<<"\nEnter the age of the patient::";**

**cin>>age;**

**}**

**voidnew\_patient::showinfo()**

**{**

**if(age<12)**

**{**

**cout<<"\nThe age of the patient is:="<<age;**

**showdata();**

**}**

**else**

**cout<<"\nOnly for patients less than 12 in age!";**

**}**

**void main()**

**{**

**clrscr();**

**new\_patient p;**

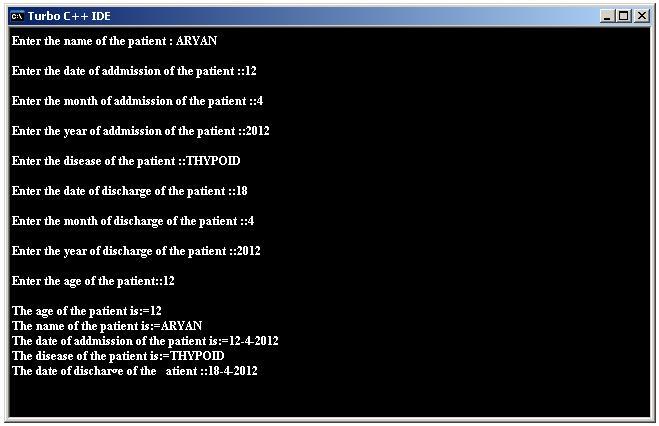
**p.getdata();**

**p.getage();**

**p.showinfo();**

**}**

**OUTPUT:-**

**[](http://1.bp.blogspot.com/-JANqF2eMCYw/T3vqaOSxc1I/AAAAAAAABH4/NCuAavxwBoo/s1600/KK.JPG)**

1. **Imagine a tollbooth with a class called toll Booth. The two data items are a type unsigned int to hold the total number of cars, and a type double to hold the total amount of money collected. A constructor initializes both these to 0. A member function called payingCar ( ) increments the car total and adds 0.50 to the cash total. Another function, called nopayCar ( ), increments the car total but adds nothing to the cash total. Finally, a member function called displays the two totals. Include a program to test this class. This program should allow the user to push one key to count a paying car, and another to count a nonpaying car. Pushing the ESC kay should cause the program to print out the total cars and total cash and then exit.**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**const char ESC=27;**

**const double TOLL=0.5;**

**class tollbooth**

**{**

**private:**

**unsignedinttotalcars;**

**doubletotalcash;**

**public:**

**tollbooth()**

**{**

**totalcars=0;**

**totalcash=0;**

**}**

**voidpayingcar()**

**{**

**totalcars+=1;**

**totalcash+=TOLL;**

**}**

**voidnopaycar()**

**{**

**totalcars+=1;**

**}**

**void display()**

**{**

**cout<<"\ncars="<<totalcars<<",cash="<<totalcash;**

**}**

**};**

**int main()**

**{**

**tollbooth booth1;**

**charch;**

**clrscr();**

**cout<<"\npress 0 for each non-paying car,"**

**<<"\n      1 for each paying car,"**

**<<"\n      Esc to exit the program.\n";**

**do**

**{**

**ch=getche();**

**if(ch=='0')**

**booth1.nopaycar();**

**if(ch=='1')**

**booth1.payingcar();**

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

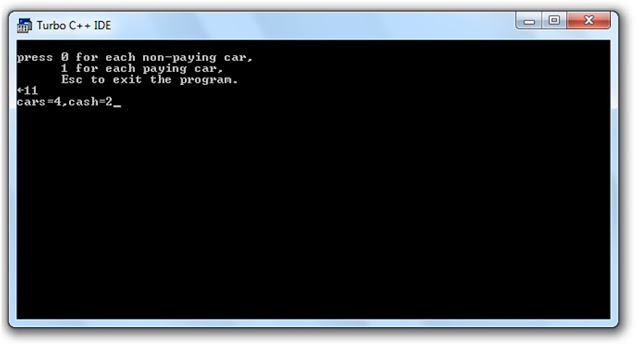
**booth1.display();**

**getch();**

**return 0;**

**}**

**OUTPUT:-**

[**[](http://1.bp.blogspot.com/-aUB2G1671SY/T2SxtzPZxZI/AAAAAAAAACw/l84XTmLpmEk/s1600/7.jpg)**](http://1.bp.blogspot.com/-aUB2G1671SY/T2SxtzPZxZI/AAAAAAAAACw/l84XTmLpmEk/s1600/7.jpg)

1. **Make a class Employee with a name and salary. Make a class Manager inherit from   
   Employee. Add an instance variable, named department, of type string. Supply a  
   method to string that prints the manager s name, department and salary. Make a class  
   Executive inherit from Manager. Supply a method to String that prints the string  
   Executive followed by the information stored in the Manager superclass object.  
   Supply a test program that tests these classes and methods**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**#include<stdio.h>**

**class employee**

**{**

**protected:**

**char nm[30];**

**floatsal;**

**};**

**classmanager:public employee**

**{**

**protected:**

**chardep[10];**

**public:**

**void input();**

**voiddisp();**

**};**

**void manager::input()**

**{**

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

**gets(nm);**

**cout<<"Enter salary: ";**

**cin>>sal;**

**cout<<"Enter department: ";**

**gets(dep);**

**}**

**void manager::disp()**

**{**

**cout<<"\nManager's name: "<<nm;**

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

**cout<<"\nDepartment: "<<dep<<endl;**

**}**

**classexecutive:public manager**

**{**

**public:**

**voiddisp(manager);**

**};**

**void executive::disp(manager m)**

**{**

**cout<<"\n\nExecutive";**

**m.disp();**

**}**

**void main()**

**{**

**clrscr();**

**manager m1;**

**m1.input();**

**executive e;**

**clrscr();**

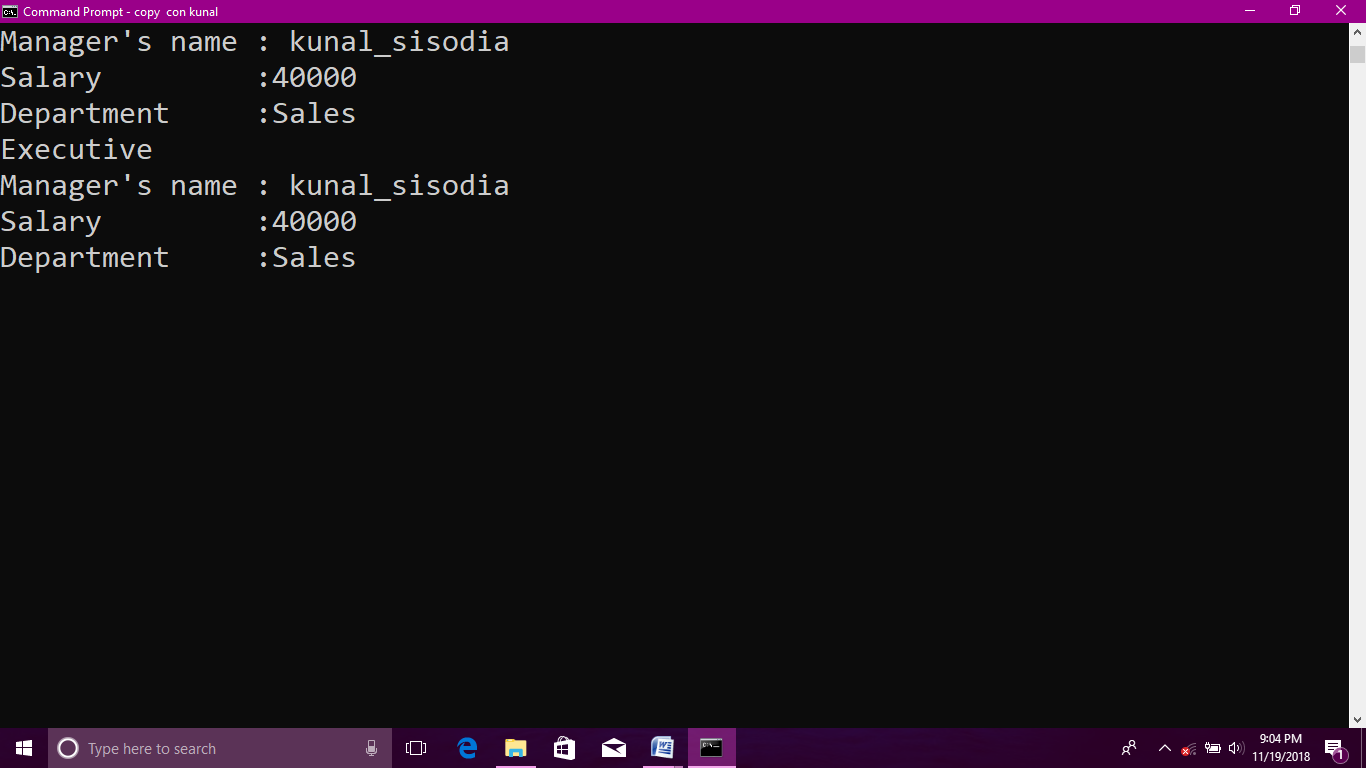
**m1.disp();**

**e.disp(m1);**

**getch();**

**}**

**OUTPUT:-**

****

1. **Write a function called reversit ( ) that reverses a string (an array of char). Use a for loop that swaps the first and last characters, then the second and next to last characters and so on. The string should be passed to reversit ( ) as an argument.   
   Write a program to exercise reversit ( ). The program should get a string from the user, call reversit ( ), and print out the result. Use an input method that allows embedded blanks. Test the program with Napoleon s famous phrase, (Able was I ere I saw Elba) .**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**#include<string.h>**

**constint MAX=80;**

**voidreversit(char[]);**

**int main()**

**{**

**charstr[MAX];**

**clrscr();**

**cout<<"\nenter the string:";**

**cin.get(str,MAX);**

**reversit(str);**

**cout<<"reversed string is:";**

**cout<<str;**

**getch();**

**return 0;**

**}**

**voidreversit(char s[])**

**{**

**intlen=strlen(s);**

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

**{**

**char temp=s[j];**

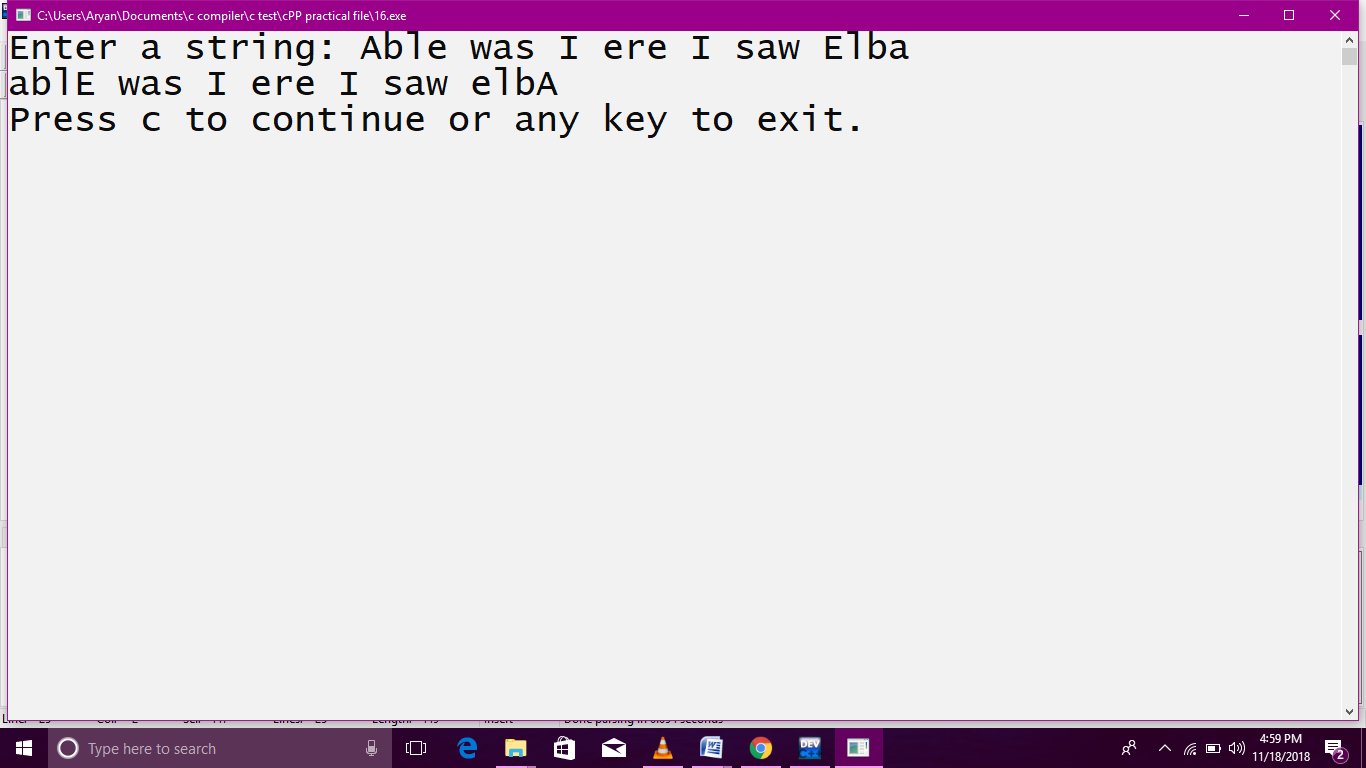
**s[j]=s[len-j-1];**

**s[len-j-1]=temp;**

**}**

**}**

**OUTPUT:-**

****

1. **Create a base class called shape. Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes called triangle and rectangle from the base shape. Add to the base class, a member function get\_data ( ) to initialize base class data members and another member function display\_area ( ) to compute and display the area of figures. Make display\_area ( ) as a virtual function and redefine this function in the derived classes to suit their requirements.Using these three classes, design a program that will accept dimensions of a triangle or a rectangle interactively and display the area. Remember the two values given as input will be treated as lengths of two sides in the case of rectangles and as base and height in the case of triangles and used as follows:**

**Area of rectangle = x \* y**

**Area of triangle = ½ \* x**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**class shape**

**{**

**protected:**

**double x;**

**double y;**

**public:**

**shape()**

**{**

**x=0.0;**

**y=0.0;**

**}**

**shape(double a,double b)**

**{**

**x=a;**

**y=b;**

**}**

**voidget\_data()**

**{**

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

**cin>>x;**

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

**cin>>y;**

**}**

**virtual void display\_area()=0;**

**};**

**classtriangle:public shape**

**{**

**public:**

**triangle()**

**{}**

**triangle(double a,double b):shape(a,b)**

**{}**

**voiddisplay\_area()**

**{**

**cout<<"Area of triangle:"<<(x\*y)/2;**

**}**

**};**

**classrectangle:public shape**

**{**

**public:**

**rectangle()**

**{}**

**rectangle(double a,double b):shape(a,b)**

**{}**

**voiddisplay\_area()**

**{**

**cout<<"Area of rectangle:"<<x\*y;**

**}**

**};**

**int main()**

**{**

**int c;**

**clrscr();**

**while(1)**

**{**

**cout<<"\n1.area of triangle";**

**cout<<"\n2.area of rectangle";**

**cout<<"\n3.exit";**

**cout<<"\nenter your choice:";**

**cin>>c;**

**switch(c)**

**{**

**case 1:triangle t;**

**t.get\_data();**

**t.display\_area();**

**break;**

**case 2:rectangle r;**

**r.get\_data();**

**r.display\_area();**

**break;**

**case 3:return 0;**

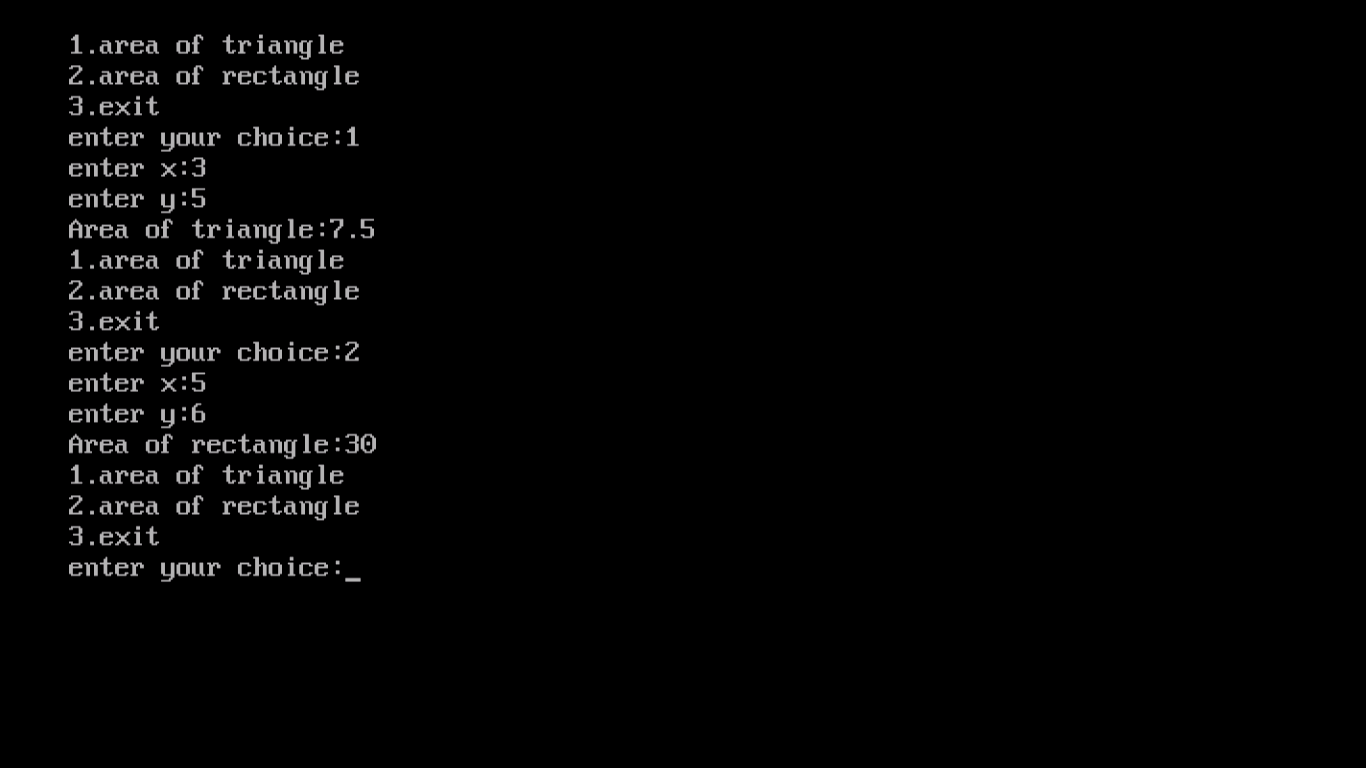
**default:cout<<"invalid choice";**

**}**

**}**

**}**

**OUTPUT:-**

****

1. **Write a program to read a file and print the number of vowels and number of words in the file. Assume that a word is a sequence of letters ending with a blank, or a tab, or an end of line marker or end of file or punctuation symbols such as “,”, ”.”, ”!” and “?”.**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**int main()**

**{**

**char line[150];**

**int vowels, consonants, digits, spaces;**

**vowels = consonants = digits = spaces = 0;**

**cout<< "Enter a line of string: ";**

**cin.getline(line, 150);**

**for(int i = 0; line[i]!='\0'; ++i)**

**{**

**if(line[i]=='a' || line[i]=='e' || line[i]=='i' ||**

**line[i]=='o' || line[i]=='u' || line[i]=='A' ||**

**line[i]=='E' || line[i]=='I' || line[i]=='O' ||**

**line[i]=='U')**

**{**

**++vowels;**

**}**

**else if((line[i]>='a'&& line[i]<='z') || (line[i]>='A'&& line[i]<='Z'))**

**{**

**++consonants;**

**}**

**else if(line[i]>='0' && line[i]<='9')**

**{**

**++digits;**

**}**

**else if (line[i]==' ')**

**{**

**++spaces;**

**}**

**}**

**cout<< "Vowels: " << vowels <<endl;**

**cout<< "Consonants: " << consonants <<endl;**

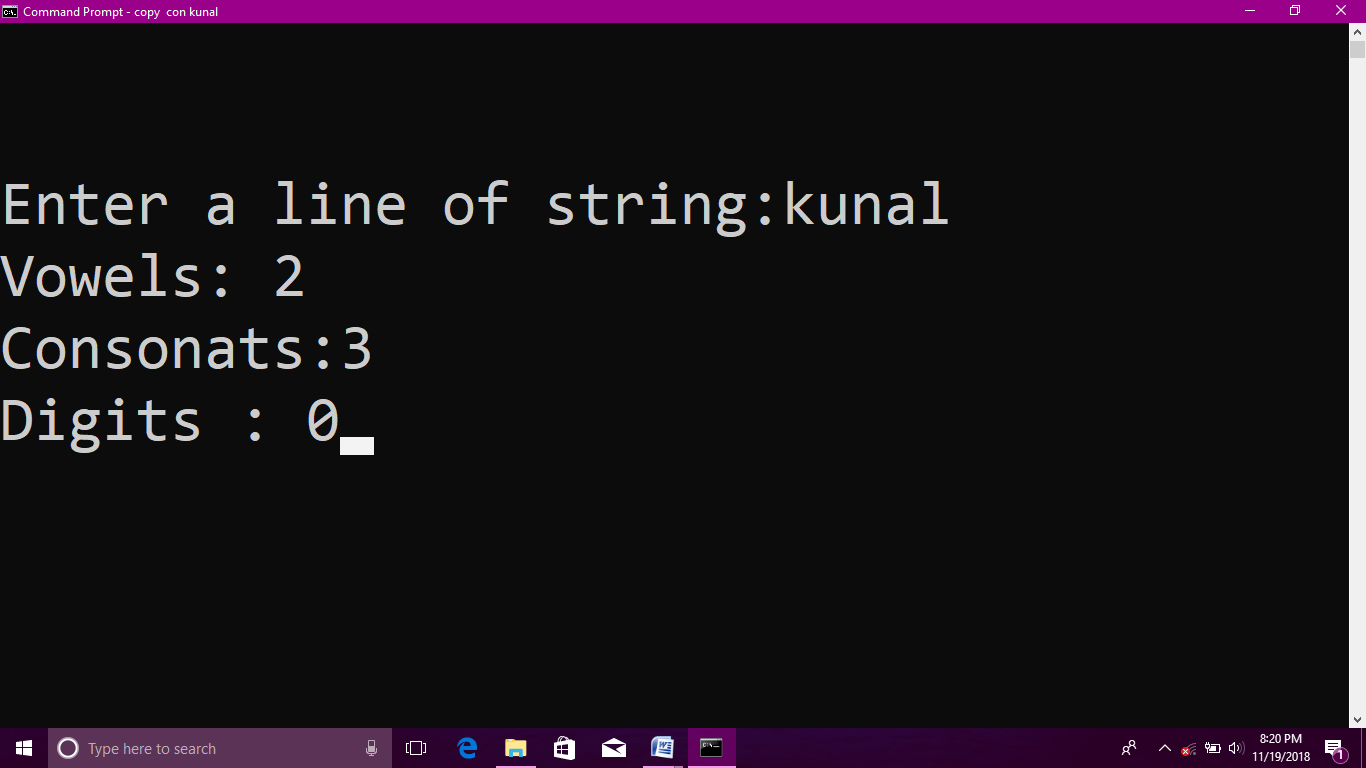
**cout<< "Digits: " << digits <<endl;**

**cout<< "White spaces: " << spaces <<endl;**

**return 0;**

**}**

**OUTPUT:-**

****

1. **Write a program to write employee detail in the file. Write a program to search give employee detain in the file.**

**PROGRAM:-**

**#include<iostream.h>**

**#include<conio.h>**

**constint m=50;**

**classemp**

**{**

**public:**

**intempno;**

**charempname[30];**

**public:**

**void get()**

**{**

**cout<<"\n Enter Employee No. : ";**

**cin>>empno;**

**cout<<"\n Enter Employee Name : ";**

**cin>>empname;**

**}**

**};**

**classfulltime:publicemp**

**{**

**public:**

**floatdaily\_rate;**

**int days;**

**int salary;**

**public:**

**voidgetdata()**

**{**

**cout<<"\n Enter Daily Rate : ";**

**cin>>daily\_rate;**

**cout<<"\n Enter No. of Days : ";**

**cin>>days;**

**}**

**voidcal()**

**{**

**salary=daily\_rate\*days;**

**cout<<"\n Salary : "<<salary;**

**}**

**void show()**

**{**

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

**cout<<"\n Employee Number : "<<empno;**

**cout<<"\n Employee Name : "<<empname;**

**cout<<"\n Salary : "<<salary;**

**cout<<"\n Status : Fulltime";**

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

**}**

**};**

**classparttime:publicemp**

**{**

**public:**

**inthourly\_rate;**

**intworking\_hours;**

**int salary1;**

**public:**

**void get1()**

**{**

**cout<<"\n Enter Hourly Rate : ";**

**cin>>hourly\_rate;**

**cout<<"\n Enter Working Hours : ";**

**cin>>working\_hours;**

**}**

**void cal1()**

**{**

**salary1=hourly\_rate\*working\_hours;**

**cout<<"\n Salary : "<<salary1<<endl;**

**}**

**void show1()**

**{**

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

**cout<<"\n Employee No : "<<empno;**

**cout<<"\n Employee Name : "<<empname;**

**cout<<"\n Salary : "<<salary1;**

**cout<<"\n Status : Part time";**

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

**}**

**};**

**int main()**

**{**

**intconstcnt=5;**

**intvar=0;**

**int var1=0;**

**fulltime f1[cnt];**

**parttime p1[cnt];**

**intx,i;**

**do**

**{**

**cout<<"\n";**

**cout<<"\n 1.Enter Record";**

**cout<<"\n 2.Display Record";**

**cout<<"\n 3.Search Record";**

**cout<<"\n 4.Quit";**

**cout<<"\n\n Enter Your Choice : ";**

**cin>>x;**

**switch(x)**

**{**

**case 1:**

**int y;**

**cout<<"\n 1. Fulltime Employee";**

**cout<<"\n 2. Parttime Employee \n";**

**cout<<"\n Enter : ";**

**cin>>y;**

**switch(y)**

**{**

**case 1:**

**f1[var].get();**

**f1[var].getdata();**

**f1[var].cal();**

**var++;**

**break;**

**case 2:**

**p1[var1].get();**

**p1[var1].get1();**

**p1[var1].cal1();**

**var1++;**

**break;**

**}**

**break;**

**case 2:**

**for(i=0; i<var; i++)**

**{**

**f1[i].show();**

**}**

**for(i=0; i<var1; i++)**

**{**

**p1[i].show1();**

**}**

**break;**

**case 3:**

**int a;**

**cout<<"\n Enter Employee No. : ";**

**cin>>a;**

**for (int i=0; i<var; i++)**

**{**

**if (f1[i].empno==a)**

**{**

**f1[i].show();**

**}**

**if(p1[i].empno==a)**

**{**

**p1[i].show1();**

**}**

**}**

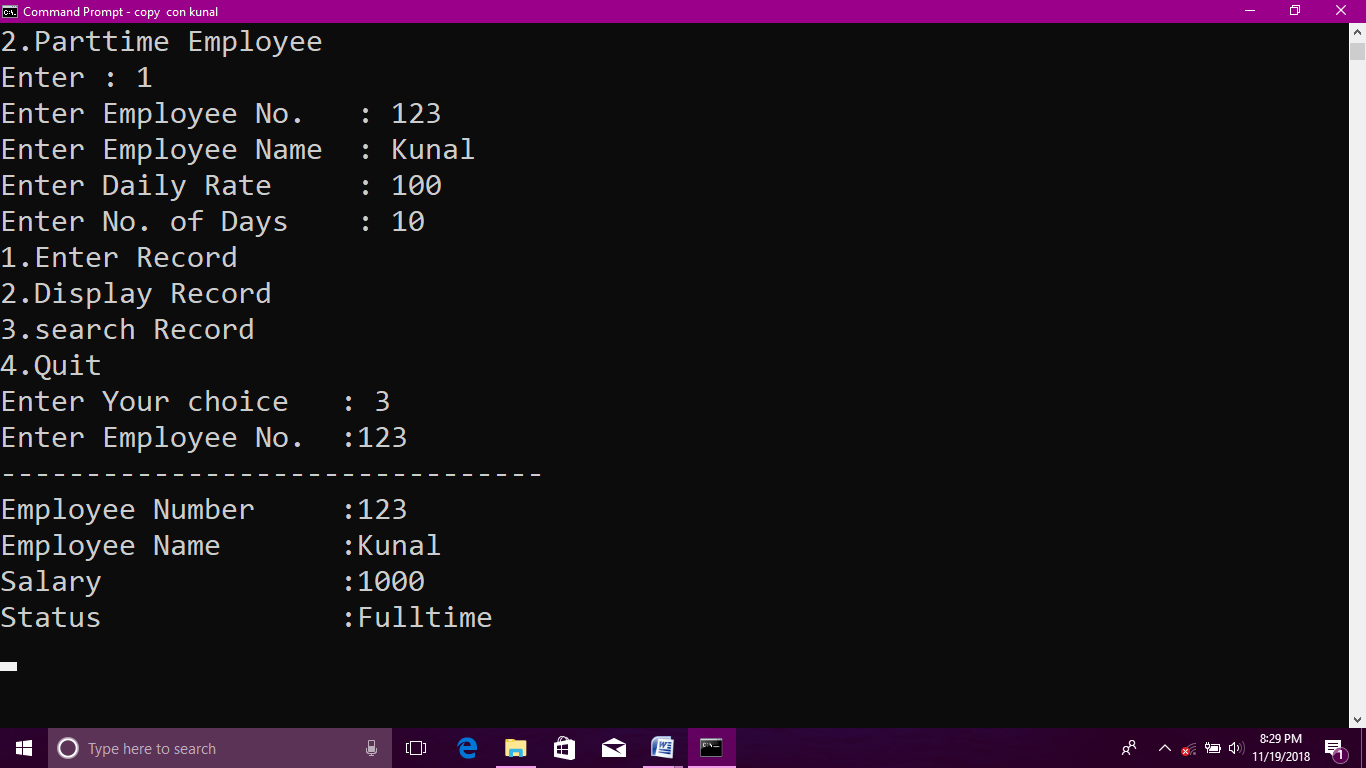
**}**

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

**return 0;**

**}**

**OUTPUT:-**

****

**19.) Write a program for array bound checking using exception handling with class concept.**

**#include <iostream>**

**#include <string>**

**class ArrayException**

**{**

**private:**

**std::string m\_error;**

**public:**

**ArrayException(std::string error)**

**: m\_error(error)**

**{**

**}**

**const char\* getError() { return m\_error.c\_str(); }**

**};**

**class IntArray**

**{**

**private:**

**int m\_data[3]; // assume array is length 3 for simplicity**

**public:**

**IntArray() {}**

**int getLength() { return 3; }**

**int& operator[](const int index)**

**{**

**if (index < 0 || index >= getLength())**

**throw ArrayException("Invalid index");**

**return m\_data[index];**

**}**

**};**

**int main()**

**{**

**IntArray array;**

**std::cout<<"Enter no. to store in array"<<std::endl;**

**try**

**{**

**for(int i=0;i<5;i++)**

**{**

**std::cin>>array[i];**

**}**

**int value = array[5];**

**}**

**catch (ArrayException &exception)**

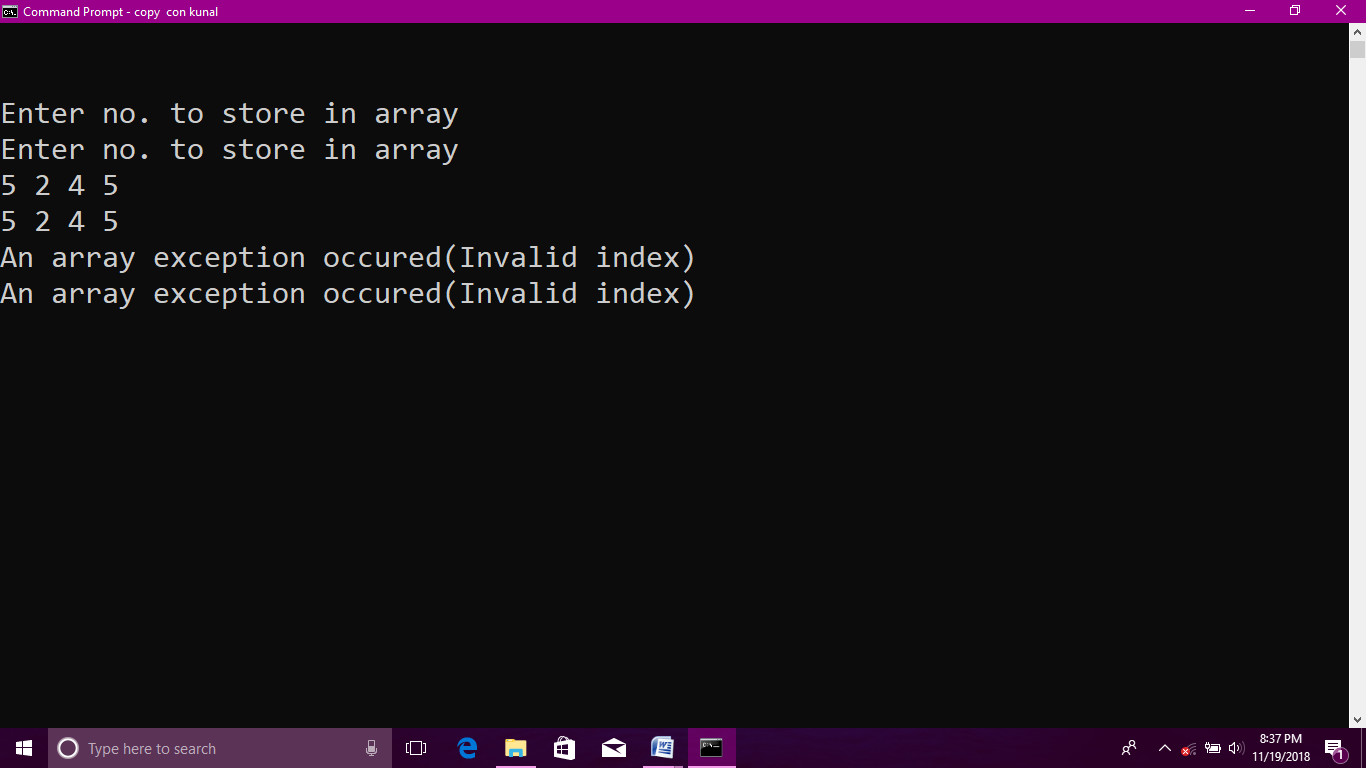
**{**

**std::cerr << "An array exception occurred (" << exception.getError() << ")\n";**

**}**

**}**

**OUTPUT:**

****

1. **Write a program to write employee detail in the file.**

**#include<iostream>**

**#include<string.h>**

**#include<stdio.h>**

**#include<cstdlib>**

**#include<fstream>**

**using namespace std;**

**char empfile[30] = "Employee.txt";**

**char ITfile[20] = "IT.txt";**

**char Adminfile[25] = "Admin.txt";**

**char Prodfile[30] = "Production.txt";**

**char Salesfile[30] = "Sales.txt";**

**class emp**

**{**

**int empid;**

**char name[30];**

**char address[60];**

**int age;**

**public:**

**char dept[15];**

**void get();**

**char \*getdept()**

**{**

**return dept;**

**}**

**};**

**void emp::get()**

**{**

**cout<<"\n Enter Employee Id : ";**

**cin>>empid;**

**cout<<"\n Enter Name : ";**

**cin>>name;**

**cout<<"\n Enter Address : ";**

**cin>>address;**

**cout<<"\n Enter department : ";**

**cin>>dept;**

**cout<<"\n Enter Age : ";**

**cin>>age;**

**}**

**void insert()**

**{**

**emp e;**

**ofstream fout; //ofstream is a class, fout is its object. It can be used only to write into the file.**

**//file is open in the binary, append and nocreate mode.**

**fout.open("Employee.txt",ios::in | ios::out | ios::binary | ios::app | ios::ate);**

**if (fout.fail())**

**{**

**cout<<"\n Unable to Open the File!!!";**

**goto err;**

**}**

**e.get(); // accepting the details from the user.**

**fout.write((char \*)&e,sizeof(e)); //writing into the file with fout object.**

**if(fout.tellp()%sizeof(e)==0)**

**{**

**cout<<"\n Record Inserted !!!"<<endl;**

**}**

**else**

**{**

**cout<<"\n Insertion Failed !!!";**

**goto err;**

**}**

**err:**

**fout.close();**

**}**

**void sort() // This function will insert the record according to department in respective file.**

**{**

**emp e;**

**ofstream adm,sal,pro,it; //all files have been created for writing mode.**

**ifstream fin; // fin object belongs to the ifstream class, it is used to read the file contents only.**

**adm.open(Adminfile, ios::out | ios::binary | ios::app);**

**sal.open(Salesfile, ios::out | ios::binary | ios::app);**

**pro.open(Prodfile, ios::out | ios::binary | ios::app);**

**it.open(ITfile, ios::out | ios::binary | ios::app);**

**fin.open(empfile, ios::in | ios::binary);**

**while(fin.read((char \*)&e,sizeof(e))) //reading the file contents till it reaches end of file.**

**{**

**if(strcmp(e.getdept(),"Admin")==0)**

**{**

**adm.write((char \*)&e,sizeof(e));**

**cout<<"\n Record Inserted into ADMIN File!!!";**

**}**

**else if(strcmp(e.getdept(),"Sales")==0)**

**{**

**sal.write((char \*)&e,sizeof(e));**

**cout<<"\n Record Inserted into SALES File!!!";**

**}**

**else if(strcmp(e.getdept(),"IT")==0)**

**{**

**it.write((char \*)&e,sizeof(e));**

**cout<<"\n Record Inserted into IT File!!!";**

**}**

**else if(strcmp(e.getdept(),"Production")==0)**

**{**

**pro.write((char \*)&e,sizeof(e));**

**cout<<"\n Record Inserted into Production File!!!";**

**}**

**else**

**cout<<"\n Insert Correct Record!!!";**

**}**

**fin.close();**

**adm.close();**

**sal.close();**

**it.close();**

**pro.close();**

**}**

**int main()**

**{**

**int n;**

**cout<<"\n Enter No. of Records You Want? : ";**

**cin>>n;**

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

**{**

**insert();**

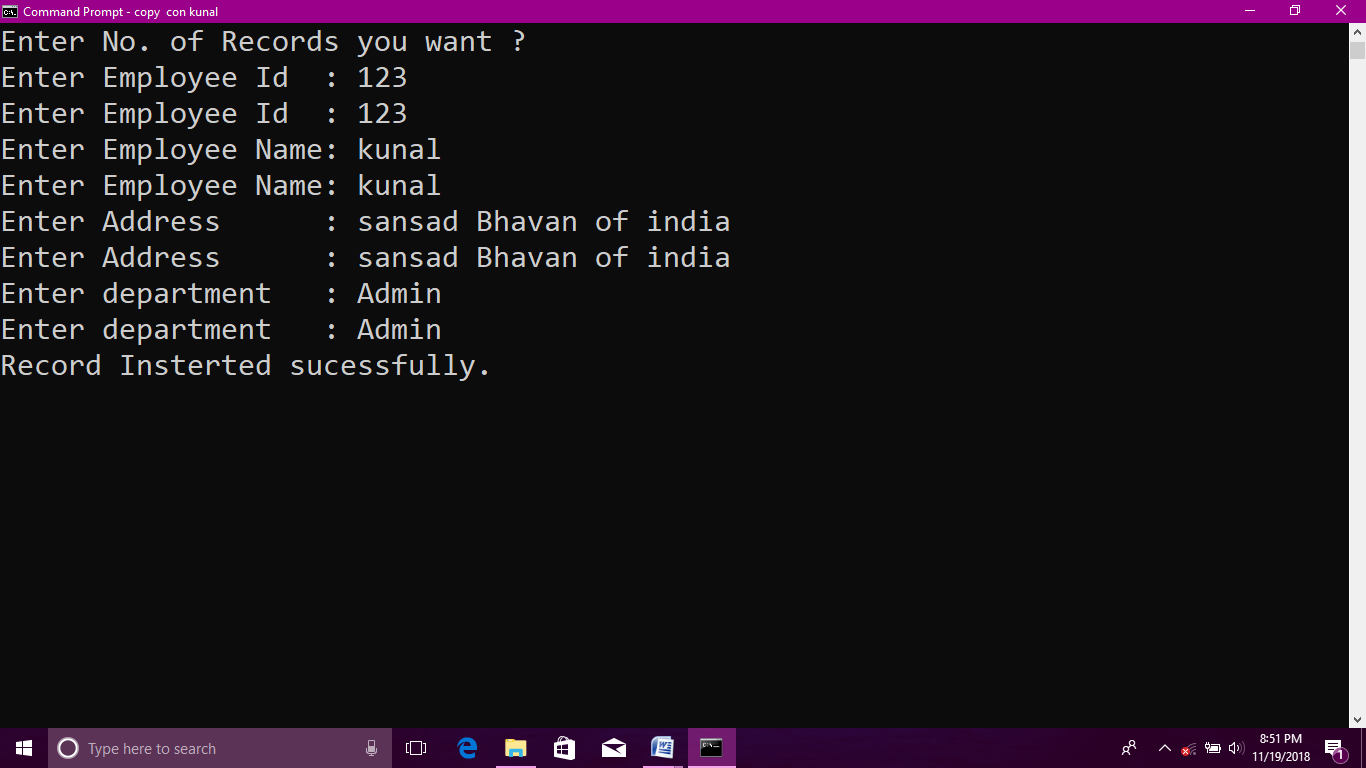
**}**

**sort();**

**return 0;**

**}**

**OUTPUT:**

****

**21.Write a program using constructor to handle divide by zero exception.**

**#include<iostream>**

**using namespace std;**

**class Divide**

**{**

**private:**

**int \*x;**

**int \*y;**

**public:**

**Divide()**

**{**

**x = new int();**

**y = new int();**

**cout<<"\nEnter two numbers: ";**

**cin>>\*x>>\*y;**

**try**

**{**

**if(\*y == 0)**

**{**

**throw \*x;**

**}**

**}**

**catch(int)**

**{**

**delete x;**

**delete y;**

**cout<<"Second number cannot be zero!"<<endl;**

**throw;**

**}**

**}**

**~Divide()**

**{**

**try**

**{**

**delete x;**

**delete y;**

**}**

**catch(...)**

**{**

**cout<<"Error while deallocating memory"<<endl;**

**}**

**}**

**float division()**

**{**

**return (float)\*x / \*y;**

**}**

**};**

**int main()**

**{**

**try**

**{**

**Divide d;**

**float res = d.division();**

**cout<<"Result of division is: "<<res;**

**}**

**catch(...)**

**{**

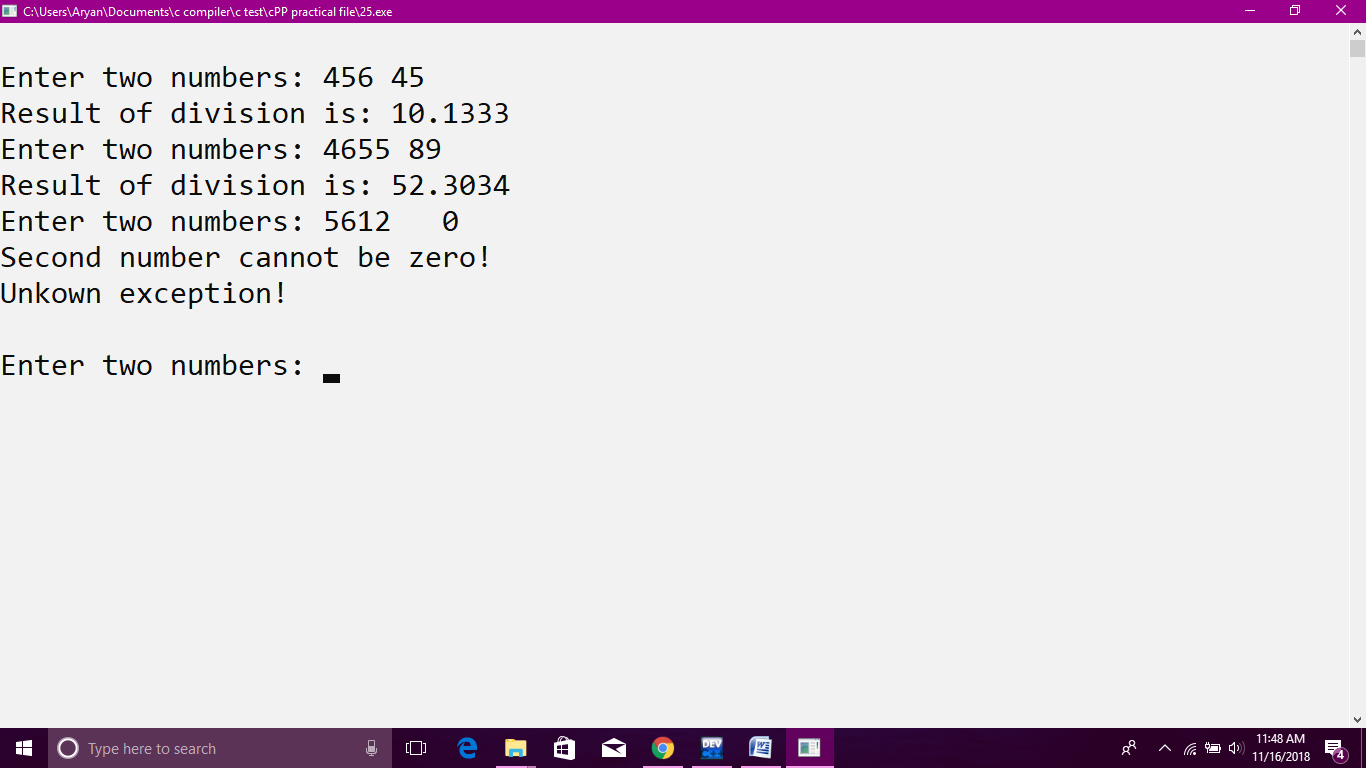
**cout<<"Unkown exception!"<<endl;**

**}**

**main();**

**return 0;**

**}**

**OUTPUT:**