Enable | Enlighten | Enrich
(Deemed to be University)
(Under Section 3 of UGC Act 1956)

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

(For the candidates admitted from 2017 onwards)

DEPARTMENT OF COMMERCE (CA)

Syllabus **Semester 3**

		L	T	P	C
17CCU312	**OBJECT ORIENTED PROGRAMMING WITH C++ (PRACTICAL)**	**-**	**-**	**4**	**2**

Scope: To enable the students to tackle complex programming problems, making good use of the object oriented programming paradigm to simplify the design and implementation process.

Objectives :

- To gain the practical knowledge on C++
- To design and implement the C++ programmes for classes operator overloading and inheritance
- To perform I/O operations on file concepts

Object and Classes

1. Create an class to implement the data structure STACK . Write a constructor to initialize the top of the STACK to zero. Write a member function PUSH () to insert an element and a member function POP () to delete an element. Check for over flow and under flow conditions

2. Create a class ARITH which consists of FLOAT and an INTEGER variable. Write member function ADD () SUB (), MUL (), DIV () MOD () to perform addition, subtraction, mulplication, division and modulus respectively. Write member functions to get and display MAT () object values.

Operator Overloading

3. Create a class "MAT" as a 2D matrix and R, C represents rows and columns of the matrix. Overload the operators +-* t add, subtract, multiply 2 matrices. Write member functions to get and display MAT () object values.

4. Create a class STRING. Write member function to initalise to get and display strings. Overload the operator + to concatenate two strings, == to compare 2 string and a member function to find the length of the strings.

Inheritance

5. Create a class which consist of EMPLOYEE details like eno, ename, dept, basic salary and grade. Write member function to get and display them. Derive a class PAY from the above class and write member function to calculate DA, HRA, PF depending on the grade and display the payslip in a neat format using console I/O.

6. Create a class SHAPE whicb consist of two virtual functions CAL_Area () and CAL_Peri () to calculate area and perimeter of various figures . Derive 3 classes SQUARE, RECTANGLE, TRIANGLE from the class SHAPE and calculate area and perimeter of each separately and display the result.

7. Create two classes, which consist of two private variables, one integer and one float variable in each class. Write member functions to get and display them. Write FRIEND function common to both classes which takes the object of the above two classes as arguments and the integer and float values of both the objects separately and display the results.

Console I/O

8. Write a user defined function USERFUN () which has the formatting commands like setw(), showpos(), precision (). Write a programme which prints a multiplication table and uses userfun() for formatting.

Files

9. Write a program to perform insertion, deletion and updation using files

10. Write a program which takes a file as arguments and copies into anpother file with line numbers using Command Line Arguments.

Suggested Readings :

Reference Books :

1. E.Balagurusamy (2013) "*Object Oriented Programming With C++*"[7th Edition]. New Delhi, Tata Mcgraw Hill Publishing Company Ltd, ,
2. Ashok N. Kamthane. (2013).*Object oriented Programming with ANSI and Turbo C++*. New Delhi, Pearson Education.
3. Chandra B. (2013) . *OOPS using C++*[2nd Edition]. New Delhi, Narosa Publishing House
4. Yashavant Kanetkar.(2013) *Let Us C++* [2nd Edition]. New Delhi, BPB Publication
5. John R . Hubbard. (2007). Programming *with C++*[2nd Edition]. New Delhi, Tata Mcgraw Hill Publishers.

Ex.No:1

OBJECT AND CLASSES: STACK OPERATION

Aim:

To create a class to implement the data structure STACKS using a constructor.

Algorithm

Step 1: Start the program.

Step 2: Define a class STACK.

Step 3: To declare the appropriate variable & PUSH, POP, DISPLAY Functions.

Step 4: Using of POP function to delete element from the STACK and check it STACK underflow or not.

Step 5: Using a PUSH function inserts an element into the STACK function and to check whether STACK overflows or not.

Step 6: Display all the elements from STACK using DISPLAY function.

Step 7: Save and Compile the program.

Step 8: Stop the program.

Program

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
#define SIZE 5
class stack
{
public:
int top,ar[SIZE],item;
public:
stack()
{
top=0;
}
void push();
void pop();
void display();
};
void stack::push()
{
if(top==SIZE)
cout<<"\nthe stack is overflow!!!";
else
{
cout<<"\nenter the element you want to push:";
cin>>item;
ar[++top]=item;
cout<<"\nelement successfully pushed in the stack!!!";
}
}
```

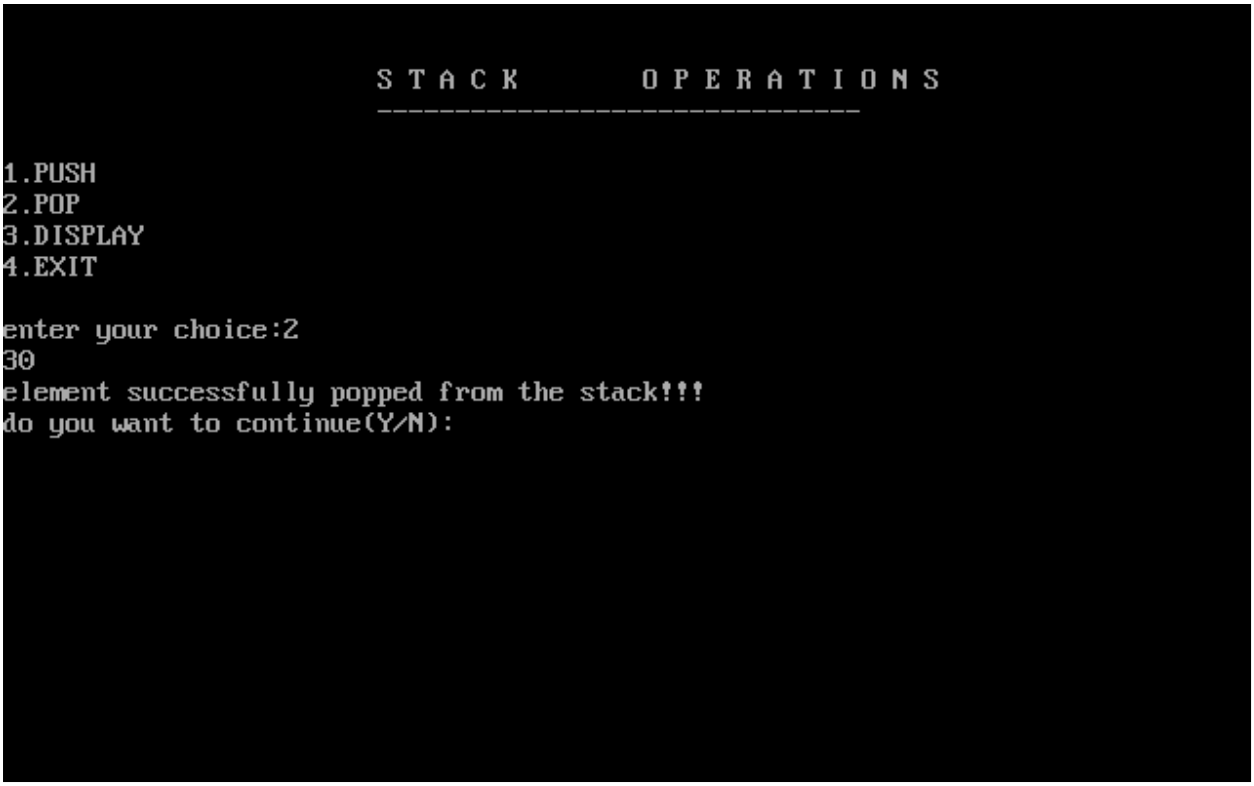
```
void stack::pop()
{
if(top==0)
cout<<"\nstack underlfow!!!";
else
{
cout<<ar[top]<<"\nelement successfully popped from the stack!!!";
top--;
}
}
void stack::display()
{
if(top<0)
cout<<"\nthe stack is empty";
else
for(int i=top;i>0;i--)
cout<<ar[i]<<" ";
}
void main()
{
char choice;
int ch,num;
stack ob;
do
{
clrscr();
cout<<"\n\n\t\tS T A C K O P E R A T I O N S";
cout<<"\n\t\t\t------------------------------";
cout<<"\n\n1.PUSH";
cout<<"\n2.POP";
cout<<"\n3.DISPLAY";
```

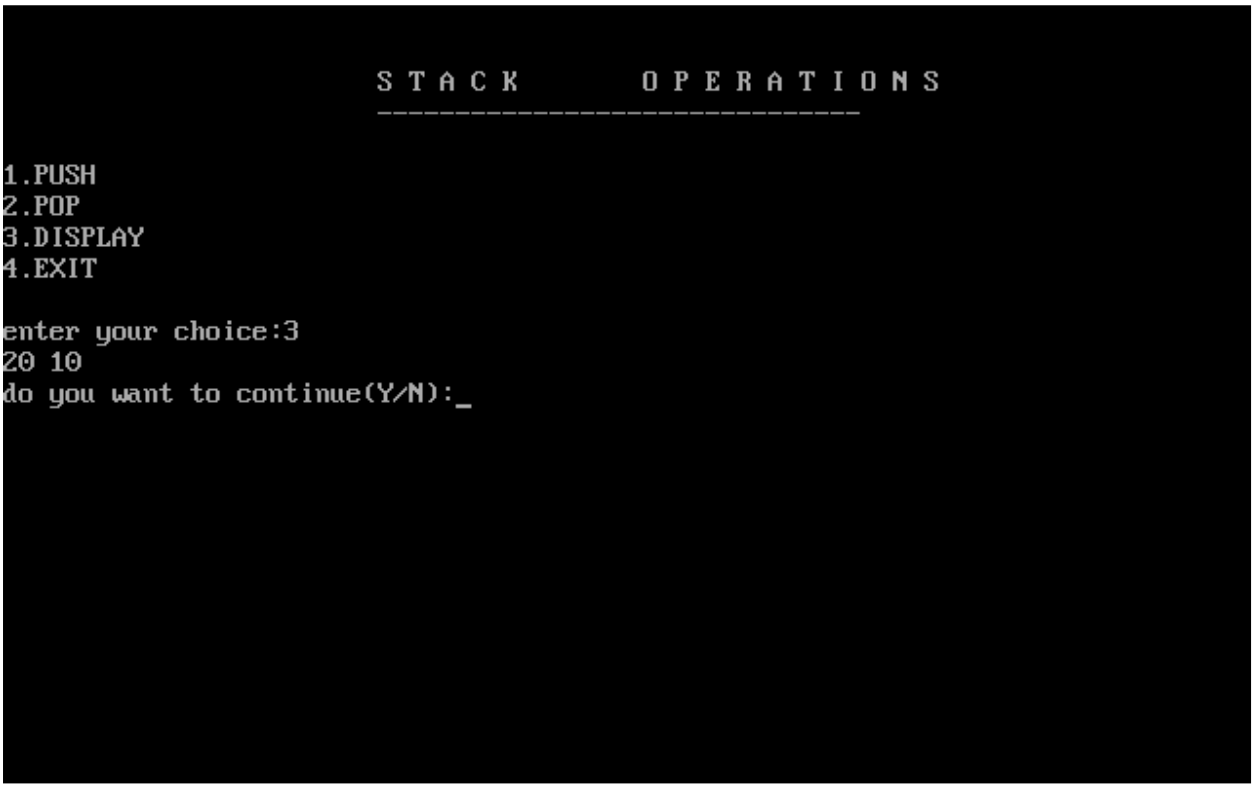
```
cout<<"\n4.EXIT";
cout<<"\n\nenter your choice:";
cin>>ch;
switch(ch)
{
case 1:
ob.push();break;
case 2:
ob.pop();break;
case 3:
ob.display();break;
case 4:
exit(0);
default:cout<<"\nplease enter the valid choice(1-4)!!!";
}
cout<<"\ndo you want to continue(Y/N):";
cin>>choice;
}
while(choice=='y'|| choice=='y');
getch();
}
```

Output:

```
                    S T A C K       O P E R A T I O N S
                    ---------------------------------

1.PUSH
2.POP
3.DISPLAY
4.EXIT

enter your choice:1

enter the element you want to push:10

element successfully pushed in the stack!!!
do you want to continue(Y/N):y
```

```
                    S T A C K       O P E R A T I O N S
                    ---------------------------------

1.PUSH
2.POP
3.DISPLAY
4.EXIT

enter your choice:3
30 20 10
do you want to continue(Y/N):_
```





Result :

The above program has been executed successfully and the output is verified.

Ex.No:2

OBJECT AND CLASSES: ARITHMETIC OPERATION

Aim:

To create a C++ program using class ARITH consists of integer and float variables.

Algorithm

Step 1: Start a program.

Step 2: Define a class ARITH and to declare appropriate variables.

Step 3: Get the values 'a' and 'b' from the user.

Step 4: To calculate the Addition, Subtraction, Multiplication, Division, and Modules using functions

Step 5: Save and Compile the program.

Step 6: Display the result.

Step 7: Stop the program.

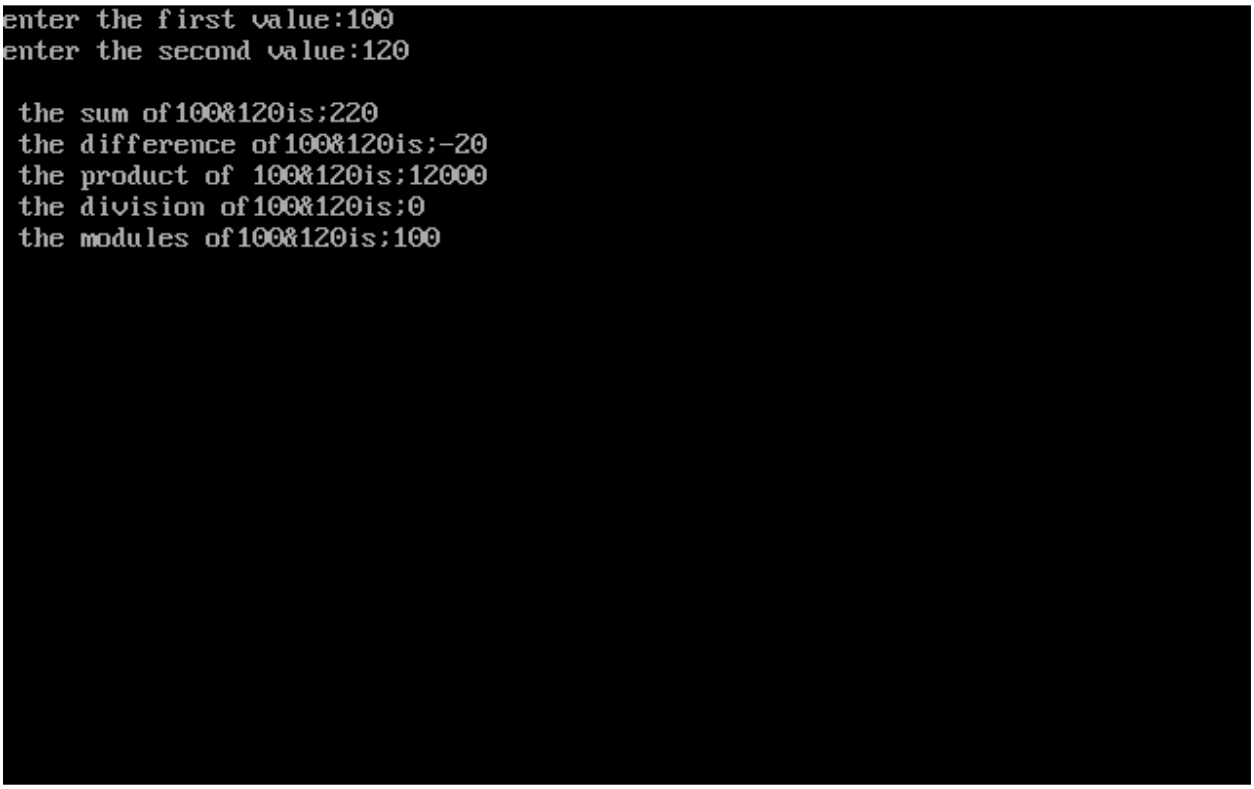
Program

```
#include<iostream.h>
#include<conio.h>
class arith
{
int a,b,ad,s,m;
float d,md;
public:
void getdata()
{
cout<<"Enter the first value:";
cin>>a;
cout<<"Enter the second value:";
cin>>b;
}
void add()
{
ad=a+b;
}
void sub()
{
s=a+b;
}
void mul()
{
m=a*b;
}
```

```
void div()
{
d=a/b;
}
void mod()
{
md=a%b;
}
void mat()
{
cout<<"\n Sum of: "<<a<<"&"<<b<<"is:"<<ad;
cout<<endl;
cout<<"\n Difference of: "<<a<<"&"<<b<<"is:"<<s;
cout<<endl;
cout<<"\n Product of "<<"&"<<b<<"is:"<<m;
cout<<endl;
cout<<"\n Division of"<<a<<"&"<<b<<"is:"<<d;
cout<<endl;
cout<<"\n Module of "<<a<<"&"<<b<<"is:"<<md;
cout<<endl;
}
};
void main()
{
clrscr();
arith z;
cout<<"Arithmetic Operations"<<endl;
cout<<"++++++++++++++++++++"<<endl;
```

```
z.getdata();
z.add();
z.sub();
z.mul();
z.div();
z.mod();
z.mat();
getch();
}
```

Output:



Result

The above program has been executed successfully and the output is verified.

Ex.No:3

OPERATOR OVERLOADING: MATRIX OPERATION

Aim:

To create a C++ program using class MAT as a 2D matrix and R, C represents Rows and Columns of the matrix.

Algorithm

Step 1: Start the program.

Step 2: Initialize the matrix using constructor function.

Step 3: Get the input values for two matrices.

Step 4: Using overloading operator +, add the two matrices and display the resultant matrix.

Step 5: Using overloading operator -, subtract the two matrices and display the resultant matrix.

Step 6: Using overloading operator *, multiply the two matrices and display the resultant matrix.

Step 7: Save and Compile the Program.

Step 8: Display the result.

Step 9: Stop the program.

Program

```
#include<iostream.h>
#include<conio.h>
class matrix
{
public:
int mat[5][5];
int row,i, j;
int column;
public:
matrix (int,int);
matrix (matrix &);
void getdata();
matrix operator-(matrix);
matrix operator+(matrix);
matrix operator*(matrix);
void display();
};
matrix::matrix(int r,int c)
{
cout<<"constructor to initialize no.of rows and columns"<<endl;
row=r;column=c;
}
matrix::matrix(matrix &m)
{
row= m.row;
column=m.column;
cout<<"copy constructor"<<endl;
for(int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
mat[i][j]=m.mat[i][j];
}}}
void matrix::getdata()
```

```
{
cout<<"value input:"<<endl;
for(int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
cout<<"value("<<i+1<<")("<<j+1<<"):";
cin>>mat[i][j];
}
}
}
matrix matrix::operator+(matrix a)
{
cout<<"addition operator"<<endl;
matrix temp(row,column);
for(int i=0;i<row;i++)
{
for( int j=0;j<column;j++)
{
temp.mat[i][j]=mat[i][j]+a.mat[i][j];
}
}
return temp;
}
matrix matrix::operator-(matrix a)
{
cout<<"subraction operator"<<endl;
matrix temp(row,column);
for (int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
temp.mat[i][j]=mat[i][j]-a.mat[i][j];
}
}
```

```
return temp;
}
matrix matrix::operator*(matrix a)
{
cout<<"multiplication operator"<<endl;
matrix temp(row,column);
for(int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
temp.mat[i][j]=0;
for(int k=0;k<column;k++)
{
temp.mat[i][j]=temp.mat[i][j]+(mat[j][k]*a.mat[k][j]);
}}}
return temp;
}
void matrix::display()
{
cout<<"the matrix is"<<endl;
for(int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
cout<<mat[i][j]<<"\t";
}
cout<<endl;
}
}
void main()
{
clrscr();
matrix m1(2,2),m2(2,2),m3(2,2);
m1.getdata();
m2.getdata();
```

```
m3=m1+m2;
m3.display();
m3=m1-m2;
m3.display();
m3=m1*m2;
m3.display();
getch();
}
```

Output:

```
value(1)(2)2
value(2)(1)2
value(2)(2)2
copy constructor
addition operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
4       5
4       5
copy constructor
subraction operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
0       1
0       1
copy constructor
multiplication operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
10      10
10      10
```

```
value(1)(2)2
value(2)(1)2
value(2)(2)2
copy constructor
addition operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
4       5
4       5
copy constructor
subraction operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
0       1
0       1
copy constructor
multiplication operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
10      10
10      10
```

Result :

The above program has been executed successfully and the output is verified.The above program has been executed successfully and the output is verified.

Ex.No:4

OPERATOR OVERLOADING: STRING OPERATION

Aim:

To create C++ program using class STRING.

Algorithm

Step 1: Start the program.

Step 2: Define a class STRING.

Step 3: Get the input values for two strings.

Step 4: Overload the operator > to check if the first string is greater than second string and print the two strings.

Step 5: Overload the operator == to compare whether the two strings are equal.

Step 6: Find the length of the string.

Step 7: Save and Compile the program.

Step 8: Stop the program.

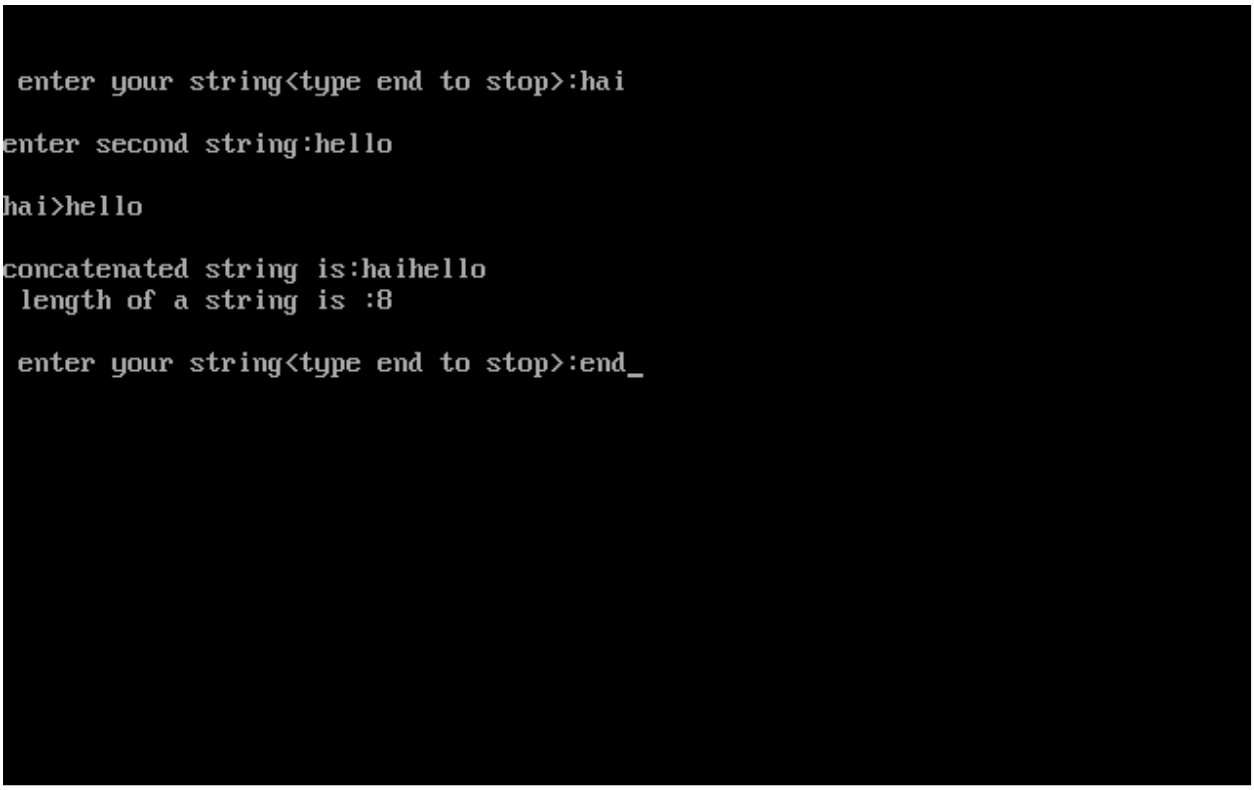
Program

```
#include<string.h>
#include<iostream.h>
#include<conio.h>
class string
{
public:
char strl[30];
string()
{
strcpy(strl,"");
}
void getstring();
void putstring();
int stlen();
int operator>(char*mm)
{
if(strcmp(strl,mm)>0)
return(1);
else
return(0);
}
int operator==(char*mm)
{
if(strcmp(strl,mm)==0)
return(1);
else
return(0);
}
char*operator+(char*sc)
{
return strcat(strl,sc);

}
};
void string::getstring()
```

```
{
cout<<endl<<"\n enter your string<type end to stop>:";
cin>>strl;
}
void string::putstring()
{
cout<<"your string is:"<<strl<<endl;
}
int string::stlen()
{
int len;
len=strlen(strl);
return(len);
}
void main()
{
string stl;
char mystr[30];
char*strcnt;
clrscr();
while(1)
{
stl.getstring();
if(stl=="end"||stl=="END"||stl=="END")
break;
cout<<endl<<"enter second string:";
cin>>mystr;
if(stl==mystr)

cout<<endl<<stl.strl<<">"<<mystr<<endl<<endl;
else
if(stl>mystr)
cout<<endl<<stl.strl<<">"<<mystr<<endl<<endl;
else
cout<<endl<<stl.strl<<">"<<mystr<<endl<<endl;
strcnt=new char[30];
```

```
strcnt=stl+mystr;
cout<<"concatenated string is:"<<strcnt;
cout<<"\n length of a string is :"<<stl.stlen();
}
}
```

Output:



Result

The above program has been executed successfully and the output is verified.

Ex.No:3

OPERATOR OVERLOADING: MATRIX OPERATION

Aim:

To create a C++ program using class MAT as a 2D matrix and R, C represents Rows and Columns of the matrix.

Algorithm

Step 1: Start the program.

Step 2: Initialize the matrix using constructor function.

Step 3: Get the input values for two matrices.

Step 4: Using overloading operator +, add the two matrices and display the resultant matrix.

Step 5: Using overloading operator -, subtract the two matrices and display the resultant matrix.

Step 6: Using overloading operator *, multiply the two matrices and display the resultant matrix.

Step 7: Save and Compile the Program.

Step 8: Display the result.

Step 9: Stop the program.

Program

```
#include<iostream.h>
#include<conio.h>
class matrix
{
public:
int mat[5][5];
int row,i, j;
int column;
public:
matrix (int,int);
matrix (matrix &);
void getdata();
matrix operator-(matrix);
matrix operator+(matrix);
matrix operator*(matrix);
void display();
};
matrix::matrix(int r,int c)
{
cout<<"constructor to initialize no.of rows and columns"<<endl;
row=r;column=c;
}
matrix::matrix(matrix &m)
{
row= m.row;
column=m.column;
cout<<"copy constructor"<<endl;
for(int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
mat[i][j]=m.mat[i][j];
}}}
void matrix::getdata()
```

```
{
cout<<"value input:"<<endl;
for(int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
cout<<"value("<<i+1<<")("<<j+1<<"):";
cin>>mat[i][j];
}
}
}
matrix matrix::operator+(matrix a)
{
cout<<"addition operator"<<endl;
matrix temp(row,column);
for(int i=0;i<row;i++)
{
for( int j=0;j<column;j++)
{
temp.mat[i][j]=mat[i][j]+a.mat[i][j];
}
}
return temp;
}
matrix matrix::operator-(matrix a)
{
cout<<"subraction operator"<<endl;
matrix temp(row,column);
for (int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
temp.mat[i][j]=mat[i][j]-a.mat[i][j];
}
}
```

```
return temp;
}
matrix matrix::operator*(matrix a)
{
cout<<"multiplication operator"<<endl;
matrix temp(row,column);
for(int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
temp.mat[i][j]=0;
for(int k=0;k<column;k++)
{
temp.mat[i][j]=temp.mat[i][j]+(mat[j][k]*a.mat[k][j]);
}}}
return temp;
}
void matrix::display()
{
cout<<"the matrix is"<<endl;
for(int i=0;i<row;i++)
{
for(int j=0;j<column;j++)
{
cout<<mat[i][j]<<"\t";
}
cout<<endl;
}
}
void main()
{
clrscr();
matrix m1(2,2),m2(2,2),m3(2,2);
m1.getdata();
m2.getdata();
```

```
m3=m1+m2;
m3.display();
m3=m1-m2;
m3.display();
m3=m1*m2;
m3.display();
getch();
}
```

Output:

```
value(1)(2)2
value(2)(1)2
value(2)(2)2
copy constructor
addition operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
4       5
4       5
copy constructor
subraction operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
0       1
0       1
copy constructor
multiplication operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
10      10
10      10
```

```
value(1)(2)2
value(2)(1)2
value(2)(2)2
copy constructor
addition operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
4       5
4       5
copy constructor
subraction operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
0       1
0       1
copy constructor
multiplication operator
constructor to initialize no. of rows & columns
copy constructor
the matrix is
10      10
10      10
```

Result :

The above program has been executed successfully and the output is verified.The above program has been executed successfully and the output is verified.

Ex.No:4

OPERATOR OVERLOADING: STRING OPERATION

Aim:

To create C++ program using class STRING.

Algorithm

Step 1: Start the program.

Step 2: Define a class STRING.

Step 3: Get the input values for two strings.

Step 4: Overload the operator > to check if the first string is greater than second string and print the two strings.

Step 5: Overload the operator == to compare whether the two strings are equal.

Step 6: Find the length of the string.

Step 7: Save and Compile the program.

Step 8: Stop the program.

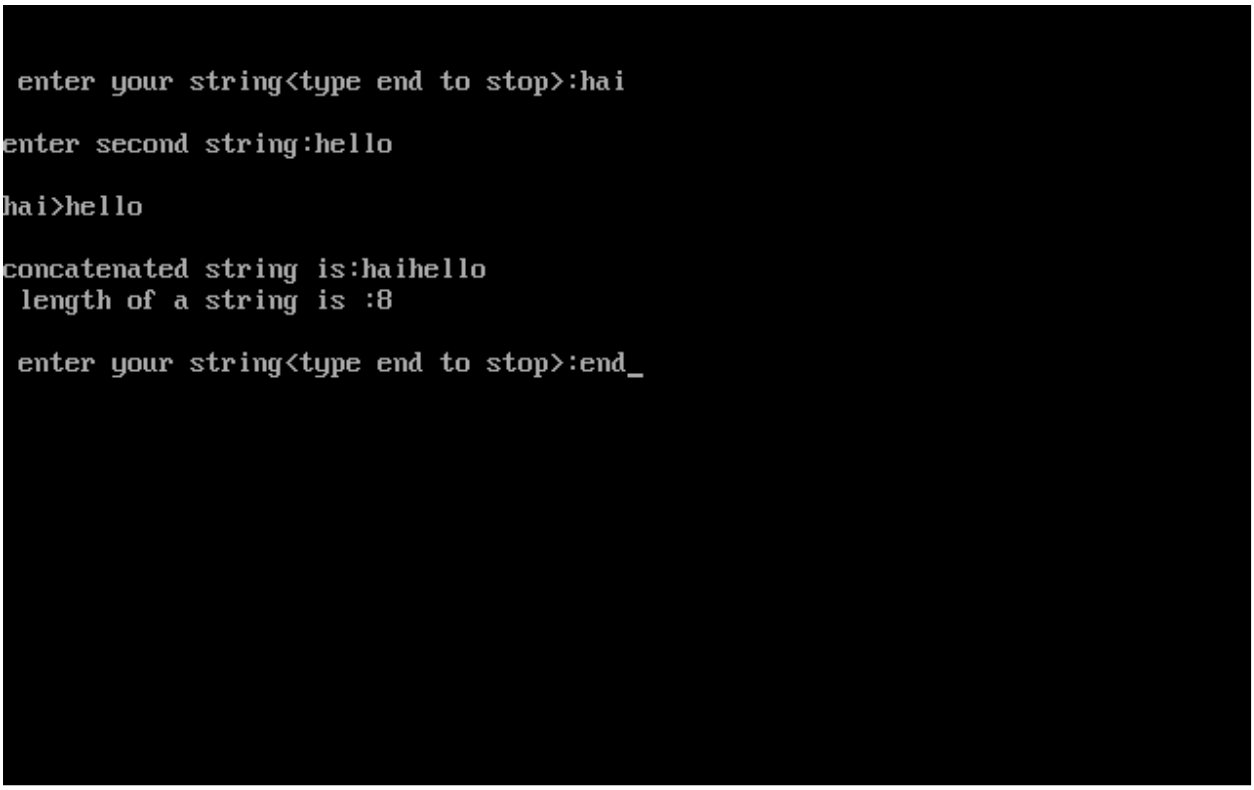
Program

```
#include<string.h>
#include<iostream.h>
#include<conio.h>
class string
{
public:
char strl[30];
string()
{
strcpy(strl,"");
}
void getstring();
void putstring();
int stlen();
int operator>(char*mm)
{
if(strcmp(strl,mm)>0)
return(1);
else
return(0);
}
int operator==(char*mm)
{
if(strcmp(strl,mm)==0)
return(1);
else
return(0);
}
char*operator+(char*sc)
{
return strcat(strl,sc);

}
};
void string::getstring()
```

```
{
cout<<endl<<"\n enter your string<type end to stop>:";
cin>>strl;
}
void string::putstring()
{
cout<<"your string is:"<<strl<<endl;
}
int string::stlen()
{
int len;
len=strlen(strl);
return(len);
}
void main()
{
string stl;
char mystr[30];
char*strcnt;
clrscr();
while(1)
{
stl.getstring();
if(stl=="end"||stl=="END"||stl=="END")
break;
cout<<endl<<"enter second string:";
cin>>mystr;
if(stl==mystr)

cout<<endl<<stl.strl<<">"<<mystr<<endl<<endl;
else
if(stl>mystr)
cout<<endl<<stl.strl<<">"<<mystr<<endl<<endl;
else
cout<<endl<<stl.strl<<">"<<mystr<<endl<<endl;
strcnt=new char[30];
```

```
strcnt=stl+mystr;
cout<<"concatenated string is:"<<strcnt;
cout<<"\n length of a string is :"<<stl.stlen();
}
}
```

Output:



Result

The above program has been executed successfully and the output is verified.

Ex.No:5

INHERITANCE: EMPLOYEE OPERATION

Aim:

To create a C++ program using class which consist of Employee Details.

Algorithm

Step 1: Start the program.

Step 2: Define a class name Employee.

Step 3: Enter the input values for employee number, employee name, department, basic salary, grade of Employee, designation.

Step 4: Calculate DA, HRA, PF, Gpay, Npay.

Step 5: Save and Compile the program.

Step 6: Display the result

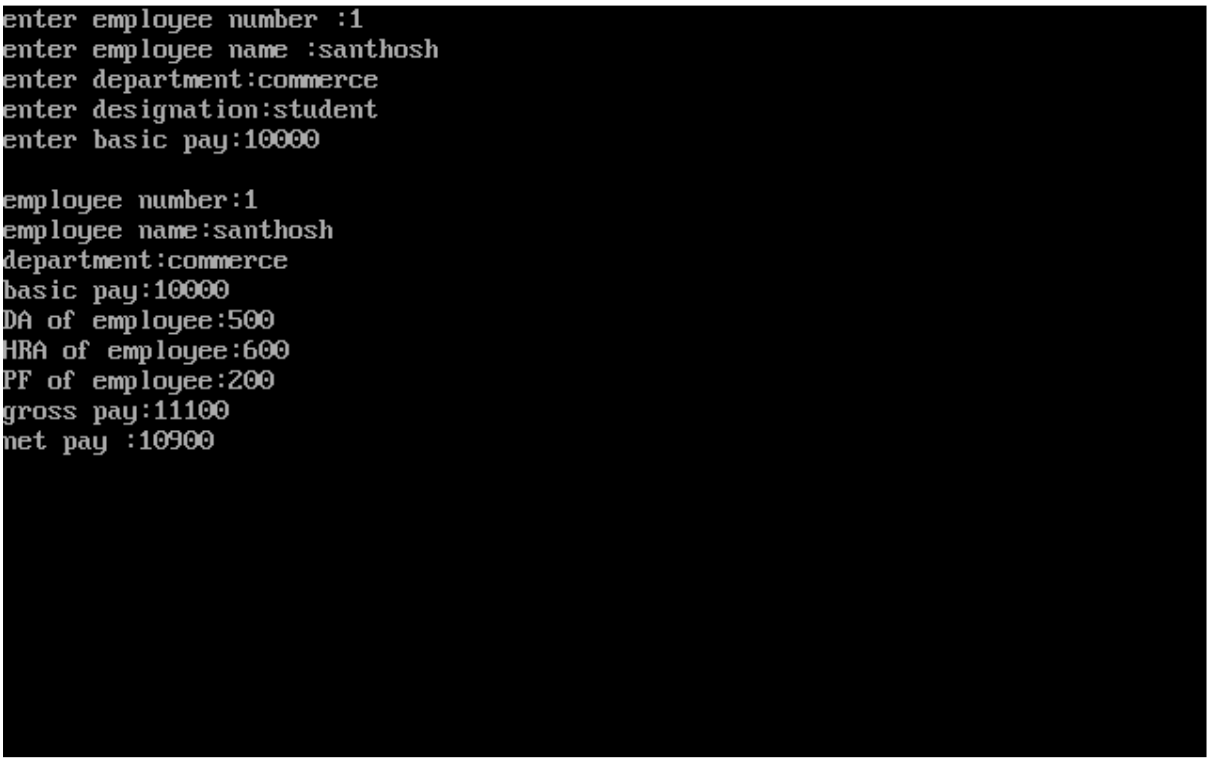
Step 7: Stop the program.

Program

```
#include<iostream.h>
#include<conio.h>
class employee
{
public:
char ename[25],dep[15],desig[15];
int eno; double bp;
void getdata()
{
cout<<"enter employee number :";
cin>>eno;
cout<<"enter employee name :";
cin>>ename;
cout<<"enter department:";
cin>>dep;
cout<<"enter designation:";
cin>>desig;
cout<<"enter basic pay:";
cin>>bp;
}
};
class payroll:public employee
{
public:
double da,hra,pf,gpay,npay;
void calc()
{
da=bp*5/100;
hra=bp*6/100;
pf=bp*2/100;
gpay=bp+da+hra;
npay=gpay-pf;
}
void display()
```

```
{
cout<<"\nemployee number:"<<eno;
cout<<"\nemployee name:"<<ename;
cout<<"\ndepartment:"<<dep;
cout<<"\nbasic pay:"<<bp;
cout<<"\nDA of employee:"<<da;
cout<<"\nHRA of employee:"<<hra;
cout<<"\nPF of employee:"<<pf;
cout<<"\ngross pay:"<<gpay;
cout<<"\nnet pay :"<<npay;
}
};
void main()
{
clrscr();
payroll e;
e.getdata();
e.calc();
e.display();
getch();
}
```

Output:



Result :

The above program has been executed successfully and the output is verified.

Ex.No:6

INHERITANCE: SHAPES

Aim:

To create a C++ program using class SHAPES which consists of two virtual function CALAREA(), CAL-PERL().

Algorithm

Step 1: Start the program.

Step 2: Create a base class called shape with 2 virtual function cal_area() and cal_peri().

Step 3: To derive sub-classes Square, Rectangle, and Triangle

Step 5: To calculate area and perimeter for shapes.

Step 6: To Save and Compile the program.

Step 7: Display the result

Step 7: Stop the program.

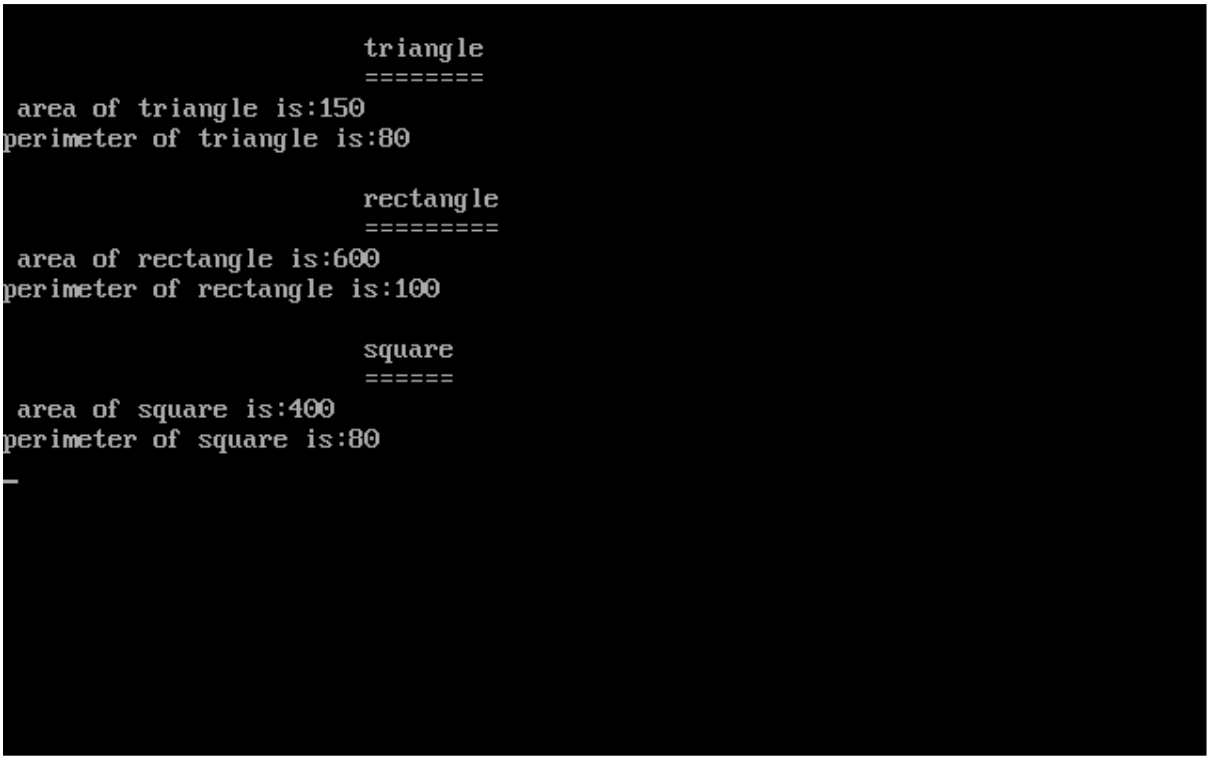
Program

```
#include<iostream.h>
#include<conio.h>
class shape
{
protected:
double l,b;
public:void getdata(double x, double y)
{
l=x;
b=y;
}
virtual void display_area(){};
virtual void display_peri(){};
};
class triangle:public shape
{
double triangle_area,triangle_peri;
void display_area()
{
triangle_area=(1*l*b)/2;
cout<<"\n area of triangle is:"<<triangle_area<<endl;
}
void display_peri()
{
triangle_peri=(2*l)+(2*b);
cout<<"perimeter of triangle is:"<<triangle_peri<<endl;
}
};
class rectangle:public shape
{
double rectangle_area,rectangle_peri;
void display_area()
{
rectangle_area=l*b;
```

```
cout<<"\n area of rectangle is:"<<rectangle_area<<endl;
}
void display_peri()
{
rectangle_peri=2*(l+b);
cout<<"perimeter of rectangle is:"<<rectangle_peri<<endl;
}
};
class square:public shape
{
double square_area,square_peri;
void display_area()
{
square_area=(l*l);
cout<<"\n area of square is:"<<square_area<<endl;
}
void display_peri()
{
square_peri=4*l;
cout<<"perimeter of square is:"<<square_peri<<endl;
}
};
void main()
{
clrscr();
shape*p;
triangle t;
rectangle r;
square s;
cout<<"\n\t\t\ttriangle";
cout<<"\n\t\t\t========";
p=&t;
p->getdata(10,30);
p->display_area();
p->display_peri();
```

```
cout<<"\n\t\t\trectangle";
cout<<"\n\t\t\t=========";
p=&r;
p->getdata(20,30);
p->display_area();
p->display_peri();
cout<<"\n\t\t\tsquare";
cout<<"\n\t\t\t======";
p=&s;
p->getdata(20,20);
p->display_area();
p->display_peri();
getch();
}
```

Output:



Result

The above program has been executed successfully and the output is verified.

Ex.No:7

CONSOLE I/O : INHERITANCE FRIEND FUNCTION

Aim:

To create a C++ program using two class which consists of two private variable one integer one float variable in each class.

Algorithm

Step 1:- Start the program.

Step 2:- Create 2 classes called “first” and “second” with 2 data members.

Step 3:- Write individual member function to get the value for 2 members.

Step 4:- Write a friend function common to both classes that is used to display the values.

Step 5:- Create separate objects for both classes and the get() function.

Step 6:- Call the display function with the objects of both classes as argument and display the two values.

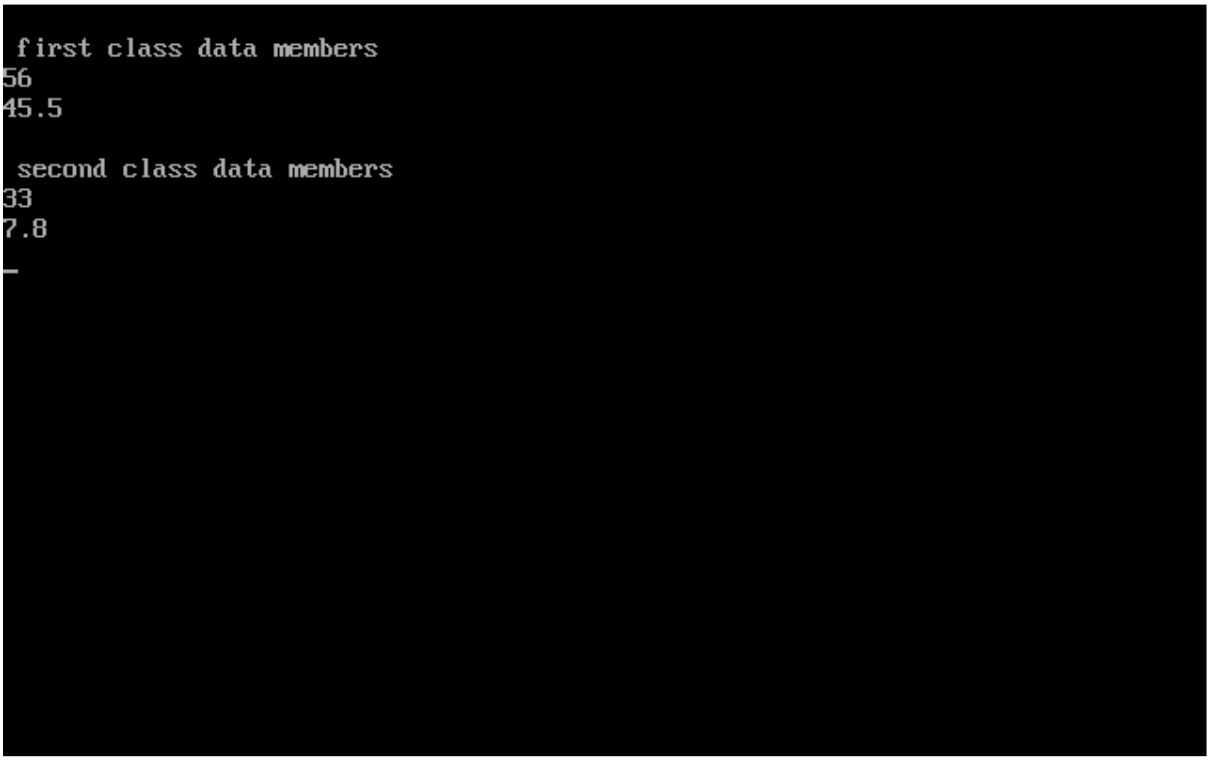
Step 7: Stop the program.

Program

```
#include<iostream.h>
#include<conio.h>
class second;
class first
{
int x;
float y;
public:
void fget(int,float);
friend void disp(first,second);
};
void first::fget(int r,float s)
{
x=r;
y=s;
}
class second
{
int a;
float b;
public:
void sget(int,float);
friend void disp(first,second);
};
void second::sget(int m,float n)
{
a=m;
b=n;
}
void disp(first f,second s)
{

cout<<"\n first class data members\n";
cout<<f.x<<endl;
cout<<f.y<<endl;
```

```
cout<<"\n second class data members\n";
cout<<s.a<<endl;
cout<<s.b<<endl;
}
void main()
{
clrscr();
first fobj;
second sobj;
fobj.fget(56,45.5);
sobj.sget(33,7.8);
disp(fobj,sobj);
getch();
}
```

Output:



Result :

The above program has been executed successfully and the output is verified.

Ex.No:8

CONSOLE I/O: MULTIPLICATION TABLE

Aim:

To write a C++ program to implement User defined function with formatting commands.

Algorithm

Step 1:- Start the program.

Step 2:- Define a class named multi and in the member function getdata, get the input values for limit & the member to multiply.

Step 3:- Define a user defined function userfun( ) to provide the decimal precision of 2.

Step 4:- Set the width of the member to 7.

Step 5:- Print the values of the number in the multiplication tables.

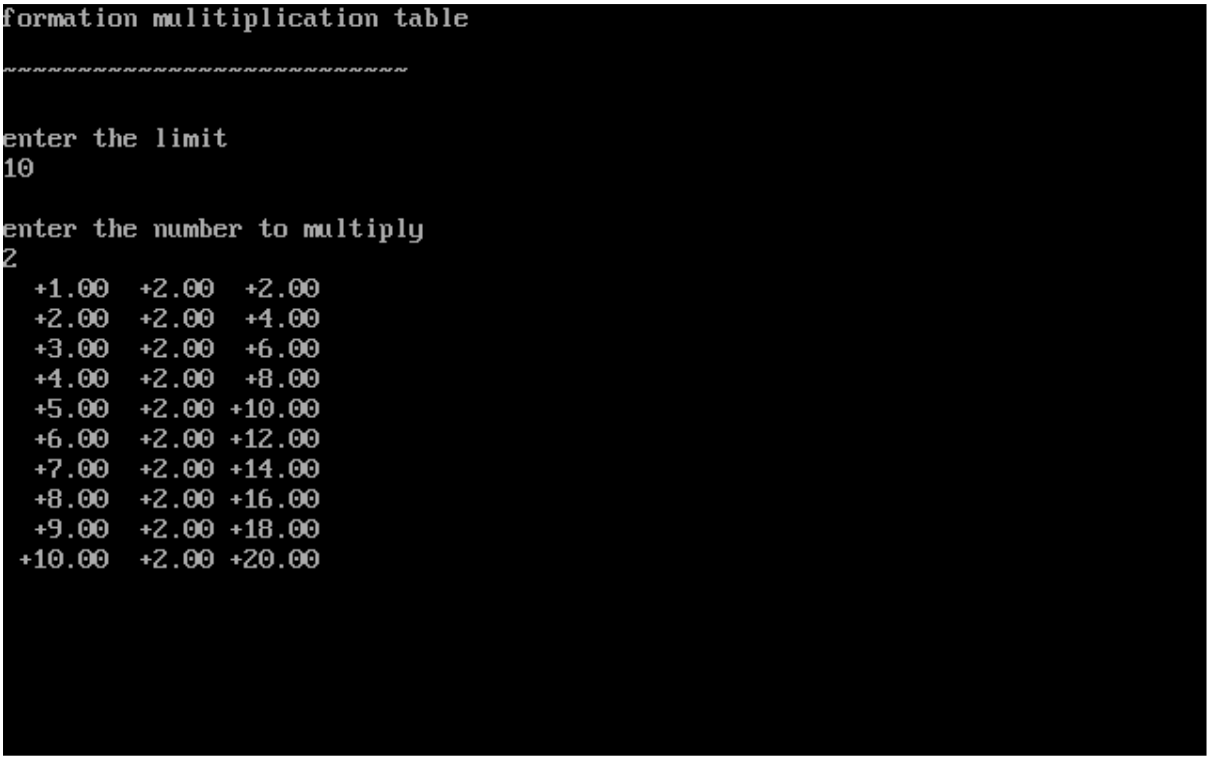
Step 6:- Stop the program.

Program

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
class multi
{
float i,l,n;
public:
void getdata()
{
cout<<"\nenter the limit\n";
cin>>l;
cout<<"\nenter the number to multiply \n";
cin>>n;
}
void userfun()
{
cout.setf(ios::showpoint);
cout.setf(ios::showpos);
cout.precision(2);
cout<<setw(7);
}
void table();
};
void multi::table()
{
for (i=1;i<=l;i++)
{
userfun();
cout<<i;
userfun();

cout<<n;
userfun();
cout<<n*i<<endl;
}
```

```
}
void main()
{
clrscr();
multi m;
cout<<"formation multiplication table\n";
cout<<"\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
m.getdata();
m.table();
getch();
}
```

Output:



Result

The above program has been executed successfully and the output is verified.

Ex.No:9

FILE : INSERTIONS, DELETION AND UPDATE

Aim:

To write a C++ program to perform insertion, deletion and updation operations using files.

Algorithm

Step 1: Start the program.

Step 2: Display the menu items to the user.

Step 3: If option=1, create a new file and get input for number of items.

Get the input for item number, name, price and quantity.

After adding the entries display them.

Step 4: If option=3, get the item number to be updated and modify the entries.

Step 5: If option=4, delete the record whose item number is given as input.

Step 6: If option=5, display records in the required format.

Step 7: Stop the program.

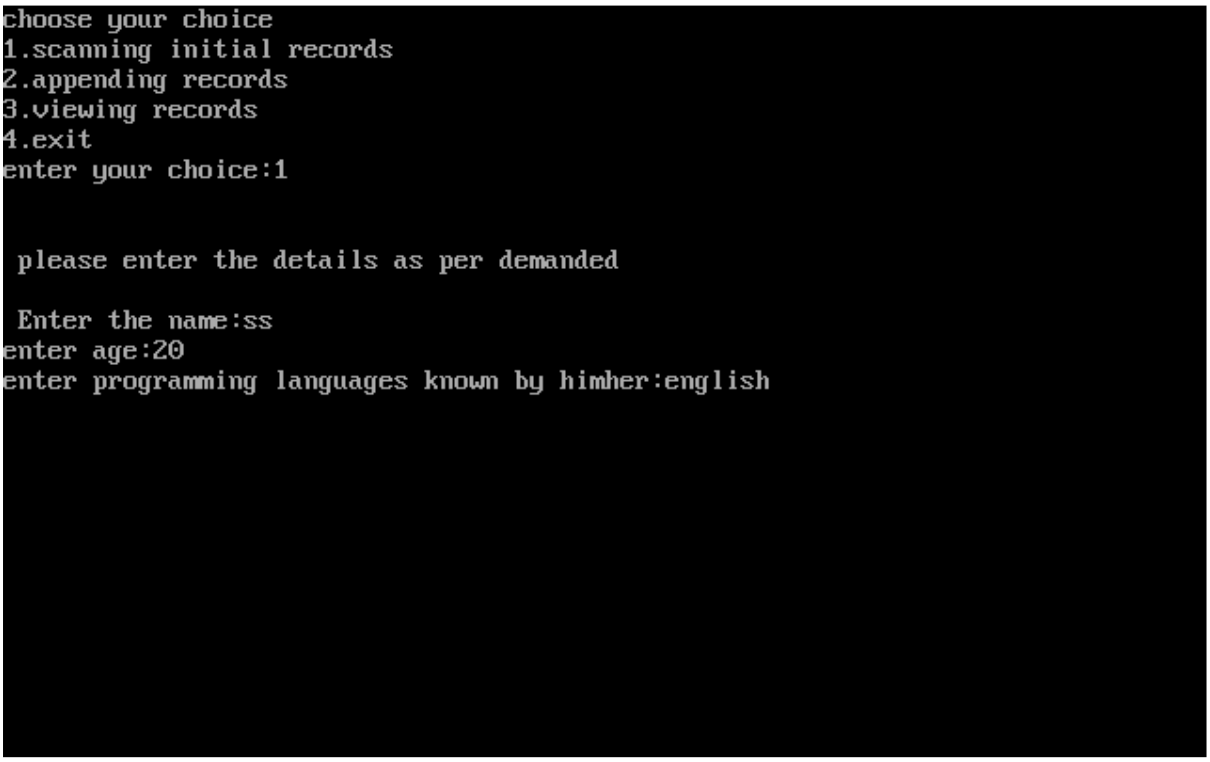
Program

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
static int totrec=0;
void main()
{
int choice;
while(1)
{
clrscr();
cout<<"choose your choice\n";
cout<<"1.scanning initial records\n";
cout<<"2.appending records\n";
cout<<"3.viewing records\n";
cout<<"4.exit\n";
cout<<"enter your choice:";
cin>>choice;
switch(choice)
{
case 1:
{
ofstream outfile;
outfile.open("emp",ios::out);
cout<<"\n\n please enter the details as per demanded\n";
cout<<"\n Enter the name:";
char name[20];
cin>>name;
outfile<<name<<endl;
cout<<"enter age:";
int age;
cin>>age;
outfile<<age<<endl;
cout<<"enter programming languages known by him\her:";
char lang[25];
```

```
cin>>lang;
outfile<<lang<<endl;
totrec=totrec+1;
outfile.close();
}
break;
case 2:
{
ofstream outfile;
outfile.open("emp",ios::app);
cout<<"\n\nplease enter the details as per demanded\n";
cout<<"\n enter the name :";
char name[20];
cin>>name;
outfile<<name<<endl;
cout<<"enter age:";
int age;
cin>>age;
outfile<<age<<endl;
cout<<"\n enter programming languages known to him \her:";
char lang[25];
cin>>lang;
outfile<<lang<<endl;
totrec=totrec+1;
outfile.close();
}
break;
case 3:
{
ifstream infile;
infile.open("emp",ios::in);
const int size=80;
char line[size];
int counter=totrec;
while(counter>=0)
```

```
{
infile.getline(line,size);
cout<<"\n\nNAME:"<<line;
infile.getline(line,size);
cout<<"\n\nAGE:"<<line;
infile.getline(line,size);
cout<<"\nLANGUAGES:"<<line;
counter--;
}
infile.close();
}
getch();
break;
case 4:
goto out;
default:cout<<"\n invalid choice\n TRY AGAIN\n";
}
}
out:
}
```

Output:



Result :

The above program has been executed successfully and the output is verified.

Ex.No:10

FILE : COMMAND LINE ARGUMENT

Aim:

To write a C++ program to copy the contents of one file to another using command line argument.

Algorithm:

Step 1:- Start the program.

Step 2:- Get arguments for filename, source and destination file to copy.

Step 3:- If source and destination file cannot be opened display error message.

Step 4:- Open the source file and read its contents character by character and copy to the destination file

Step 5:- Open the destination file and print the contents along with line number

Step 6:- Close the source and destination file

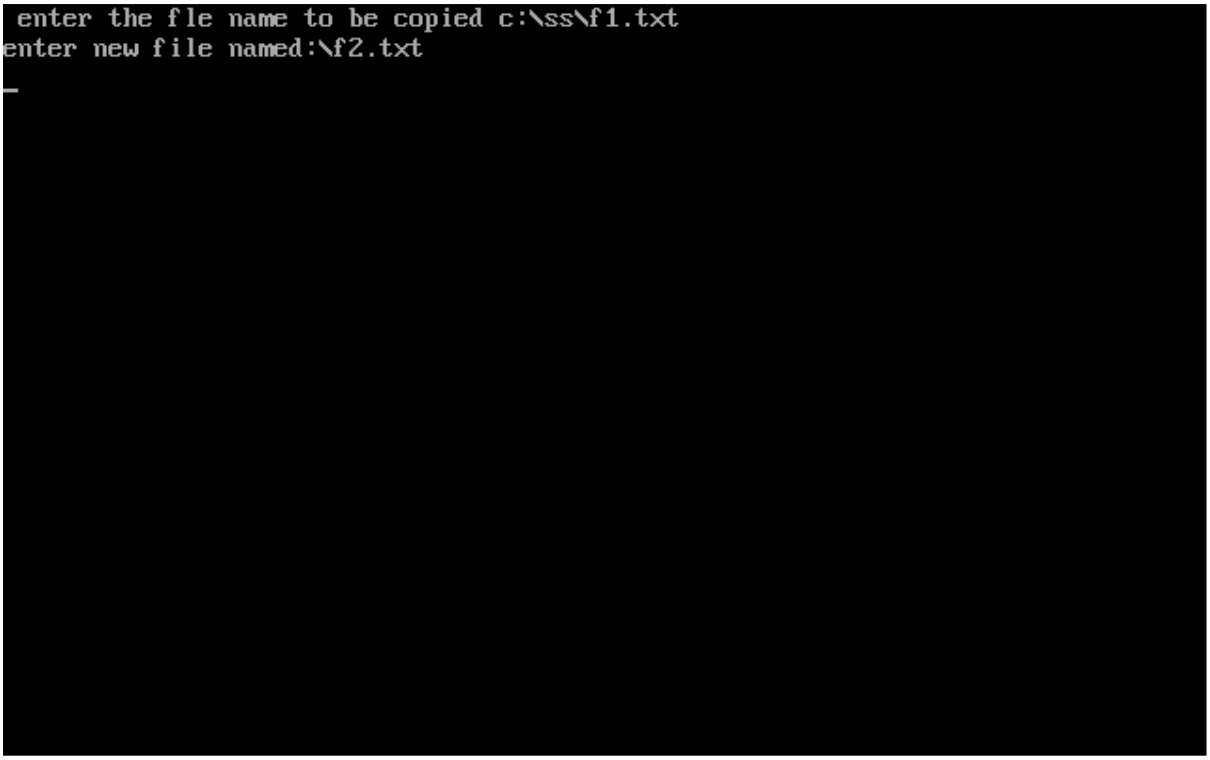
Step 7:- End the program.

Program

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>
#include<ctype.h>
#include<fstream.h>
void main()
{
ofstream outfile;
ifstream infile;
char fname1[10],fname2[20];
char ch,uch;
clrscr();
cout<<" enter the fle name to be copied ";
cin>>fname1;
cout<<"enter new file name";
cin>>fname2;
infile.open(fname1);
if(infile.fail())
{
cerr<<"\n no such a file exit";
getch();
exit(1);
}
outfile.open(fname2);
if(outfile.fail())
{
cerr<<"unable to create a file ";
getch();
exit(1);
}
while(!infile.eof())
{
ch=(char)infile.get();
```

```
uch=toupper(ch);
outfile.put(uch);
}
infile.close();
outfile.close();
getch();
}
```

Output:



Input file:
C:\ss\f1.txt
C++ is a high level Programming language.
Output file:
D:\f2.txt

Result

The above program has been executed successfully and the output is verified.