DWARAKA DOSS GOVERDHAN DOSS VAISHNAV COLLEGE(AUTONOMOUS)

ARUMBAKKAM, CHENNAI-600 106.

DEPARTMENT OF COMPUTER SCIENCE (UG & PG)

OBJECT ORIENTED PROGRAMMING USING C++ LAB

DWARAKA DOSS GOVERDHAN DOSS VAISHNAV COLLEGE(AUTONOMOUS)

ARUMBAKKAM, CHENNAI-600 106.

DEPARTMENT OF COMPUTER SCIENCE (UG & PG)

OBJECT ORIENTED PROGRAMMING USING C++ LAB

This is to certifiy that ……..... Reg no …….. of I B.Sc Computer Science has completed the record work during the academic year 2024-2025.

Faculty In-Charge **Head In-Charge**

Submitted for the practical examination held on.....at Dwaraka Doss Goverdhan Doss Vaishnav College, Chennai-106.

Internal Examiner **External Examiner**

TABLE OF CONTENTS

Ex no:01 **INLINE FUNCTION WITHOUT CLASS**

Date:

AIM:

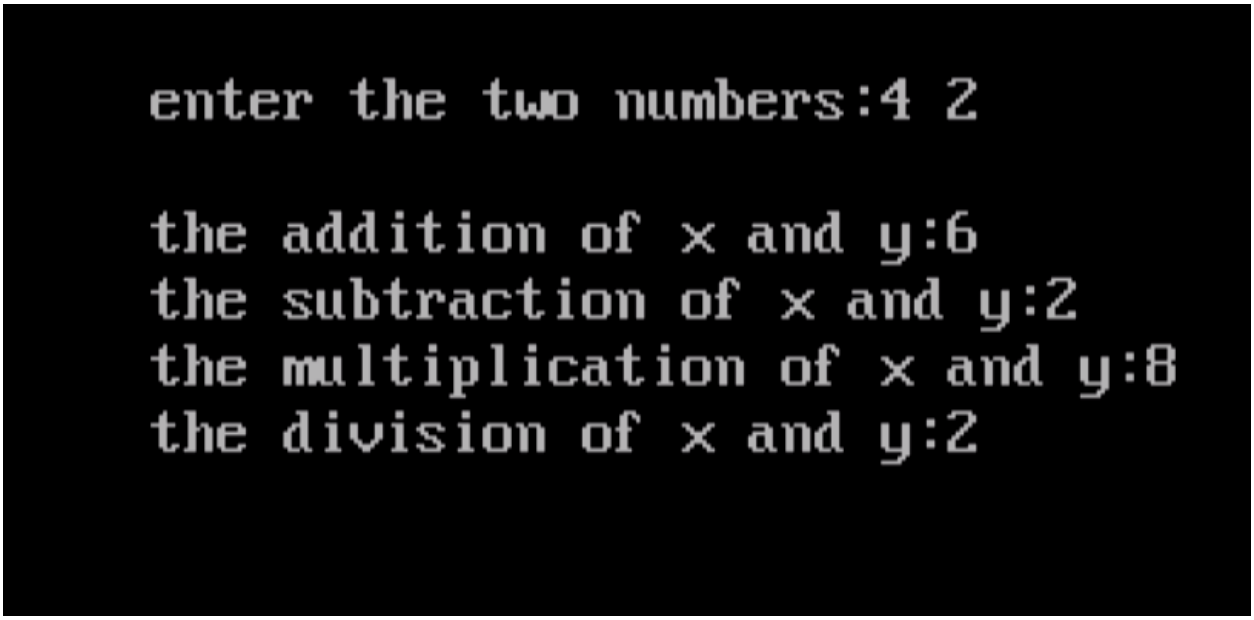
Write a program using c++ in inline function to perform arithmetic operations

CODING:

```
#include<iostream.h>
#include<conio.h>
inline float sum(float x,float y)
{
return(x+y);
}
inline float sub(float x,float y)
{
return(x-y);
}
inline float mul(float x,float y)
{
return(x*y);
}
inline double div(double x,double y)
{
return(x/y);
}
int main()
{
clrscr();
```

```
float x,y;
cout<<"\nenter the two numbers:";
cin>>x>>y;
cout<<"\nthe addition of x and y:"<<sum(x,y);
cout<<"\nthe subtraction of x and y:"<<sub(x,y);
cout<<"\nthe multiplication of x and y:"<<mul(x,y);
cout<<"\nthe division of x and y:"<<div(x,y);
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus, the c++ program to perform arithmetic operation using inline function is implemented and verified successfully

Ex no:02 FUNCTION OVERLOADING WITHOUT CLASS

Date:

AIM:

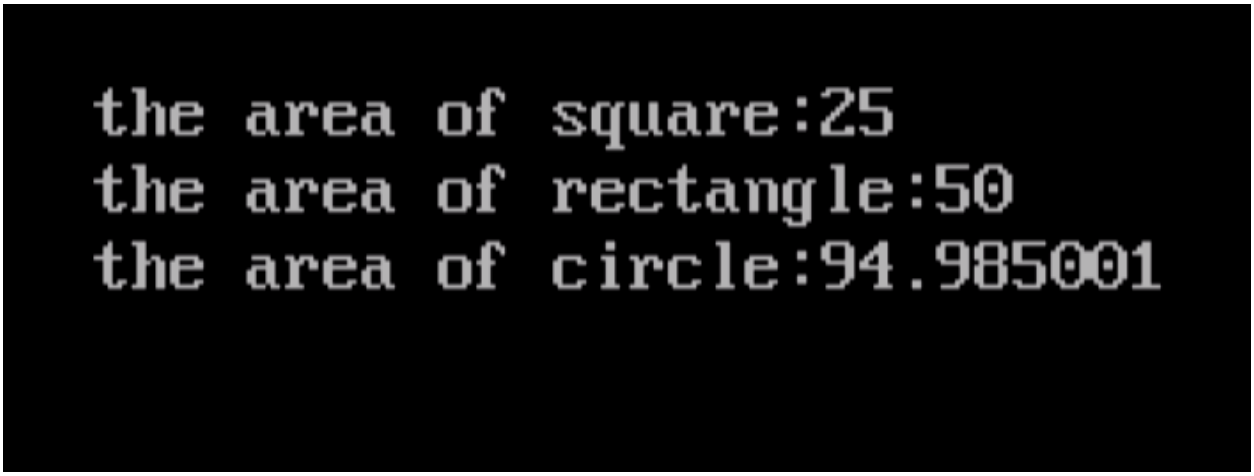
To write a c++ program for function overloading without using class to find the area of square,rectangle and circle.

CODING:

```
#include<iostream.h>
#include<conio.h>
int area(int);
int area(int,int);
float area(float);
void main()
{
clrscr();
cout<<"\nthe area of square:";
cout<<area(5);
cout<<"\nthe area of rectangle:";
cout<<area(5,10);
cout<<"\nthe area of circle:";
cout<<area(5.5f);
getch();
}
int area(int side)
{
return(side*side);
}
```

```
int area(int length,int breadth)
{
return(length*breadth);
}
float area(float radius)
{
return(3.14*radius*radius);
}
```

OUTPUT:



RESULT:

thus, the c++ program to perform function overloading implemented and verified successfully

Ex no:03 **CLASSES AND OBJECTS**

Date:

AIM:

To write a c++ program to display student mark statement using classes and objects.

CODING:

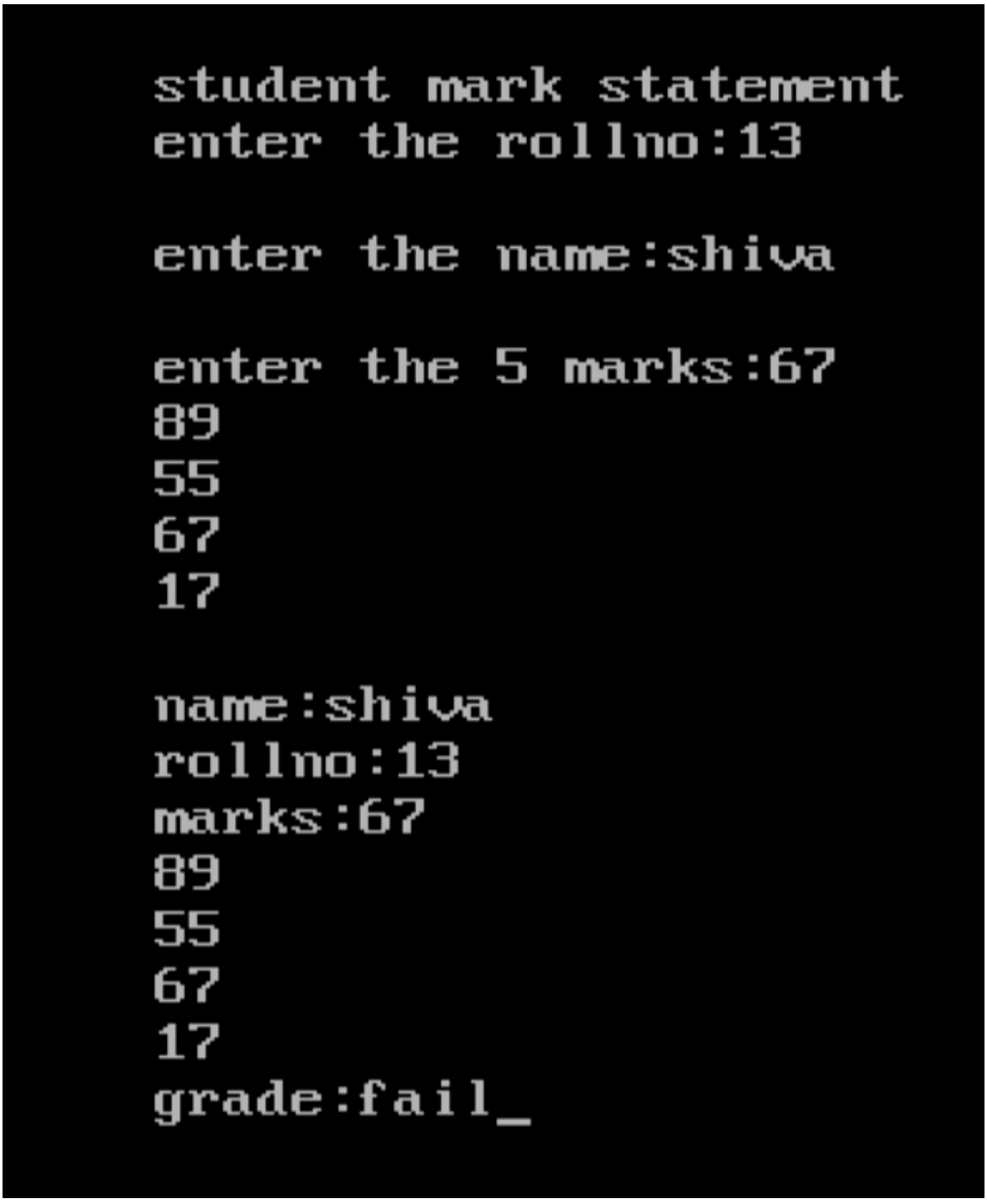
```
#include<iostream.h>
#include<string.h>
#include<conio.h>
class stud{
int rollno,m1,m2,m3,m4,m5,total;
float avg;
char name[20],grade[20];
public:void getdata();
void calculate();
void display();
};void stud::getdata(){
cout<<"\nenter the rollno:";
cin>>rollno;
cout<<"\nenter the name:";
cin>>name;
cout<<"\nenter the 5 marks:";cin>>m1>>m2>>m3>>m4>>m5;
}
void stud::calculate()
{
if((m1>40)&&(m2>40)&&(m3>40)&&(m4>40)&&(m5>40))
{
```

```
total=m1+m2+m3+m4+m5;
avg=total/5;
if(avg>=60)
strcpy(grade,"\nfirst class");
else if((avg>50)&&(avg<=60))strcpy(grade,"\nsecond class");
else strcpy(grade,"\nthird class");
}
}
void stud::display()
{
cout<<"\nname:"<<name;cout<<"\nrollno:"<<rollno;
cout<<"\nmarks:"<<m1<<"\n"<<m2<<"\n"<<m3<<"\n"<<m4<<"\n"<<m5;
if((m1>40)&&(m2>40)&&(m3>40)&&(m4>40)&&(m5>40))
{
cout<<"\ntotal:"<<total;
cout<<"\naverage:"<<avg;
cout<<"\ngrade:"<<grade;
}
else
cout<<"\ngrade:fail";
}
int main()
{
stud s;
cout<<"\nstudent mark statement";
s.getdata();
s.calculate();
s.display();
return 0;}
```

OUTPUT 1:

```
student mark statement
enter the rollno:12

enter the name:gokul

enter the 5 marks:45
67
89
64
90

name:gokul
rollno:12
marks:45
67
89
64
90
total:355
average:71
grade:
first class
```

OUTPUT 2:



RESULT:

Thus, the c++ program to display student mark statement using classes and objects is implemented and verified successfully.

Ex no:04 **FUNCTION OVRELOADING**

Date: **WITH CLASS**

AIM:

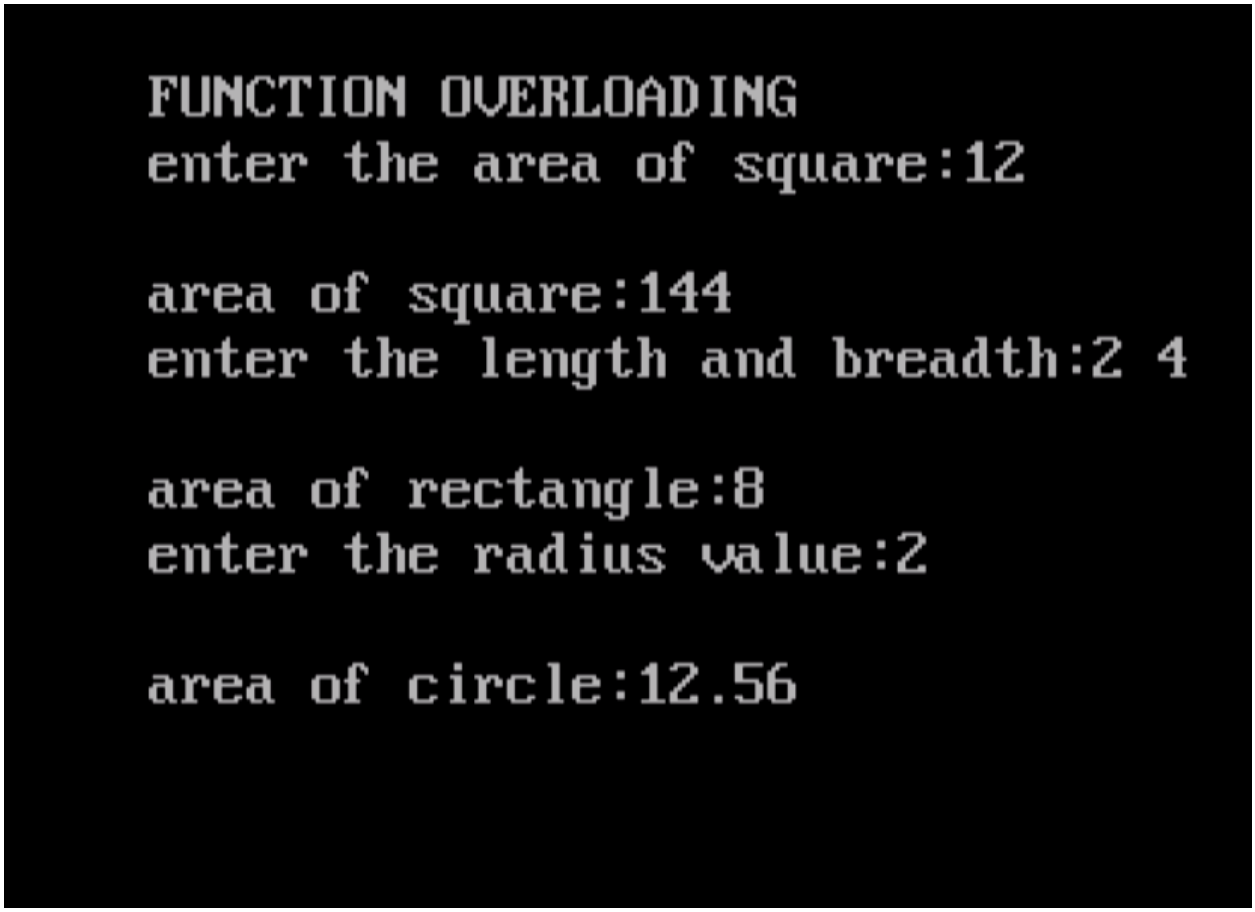
To write a c++ program to find area of square,rectangle and circle using function overloading with class.

CODING:

```
#include<iostream.h>
#include<conio.h>
class over
{
int side;
float l,b,r;
public:
void area(int);
void area(float,float);
void area(float);
};
void over::area(int side)
{
cout<<"\nenter the area of square:";
cin>>side;
cout<<"\narea of square:"<<side*side;
}
void over::area(float l,float b)
{
cout<<"\nenter the length and breadth:";
cin>>l>>b;
cout<<"\narea of rectangle:"<<l*b;
}
void over::area(float r)
{
cout<<"\nenter the radius value:";
cin>>r;
cout<<"\narea of circle:"<<3.14*r*r;
}
void main()
{
clrscr();
int side;
float l,b,r;
```

```
cout<<"\t\t\nFUNCTION OVERLOADING";
over o;
o.area(side);
o.area(l,b);
o.area(r);
getch();
}
```

OUTPUT:



RESULT:

Thus, the c++ program to perform function overloading with class is implemented and verified successfully.

Ex no:05 FRIEND FUNCTION SWAPPING OF TWO VALUES

Date:

AIM:

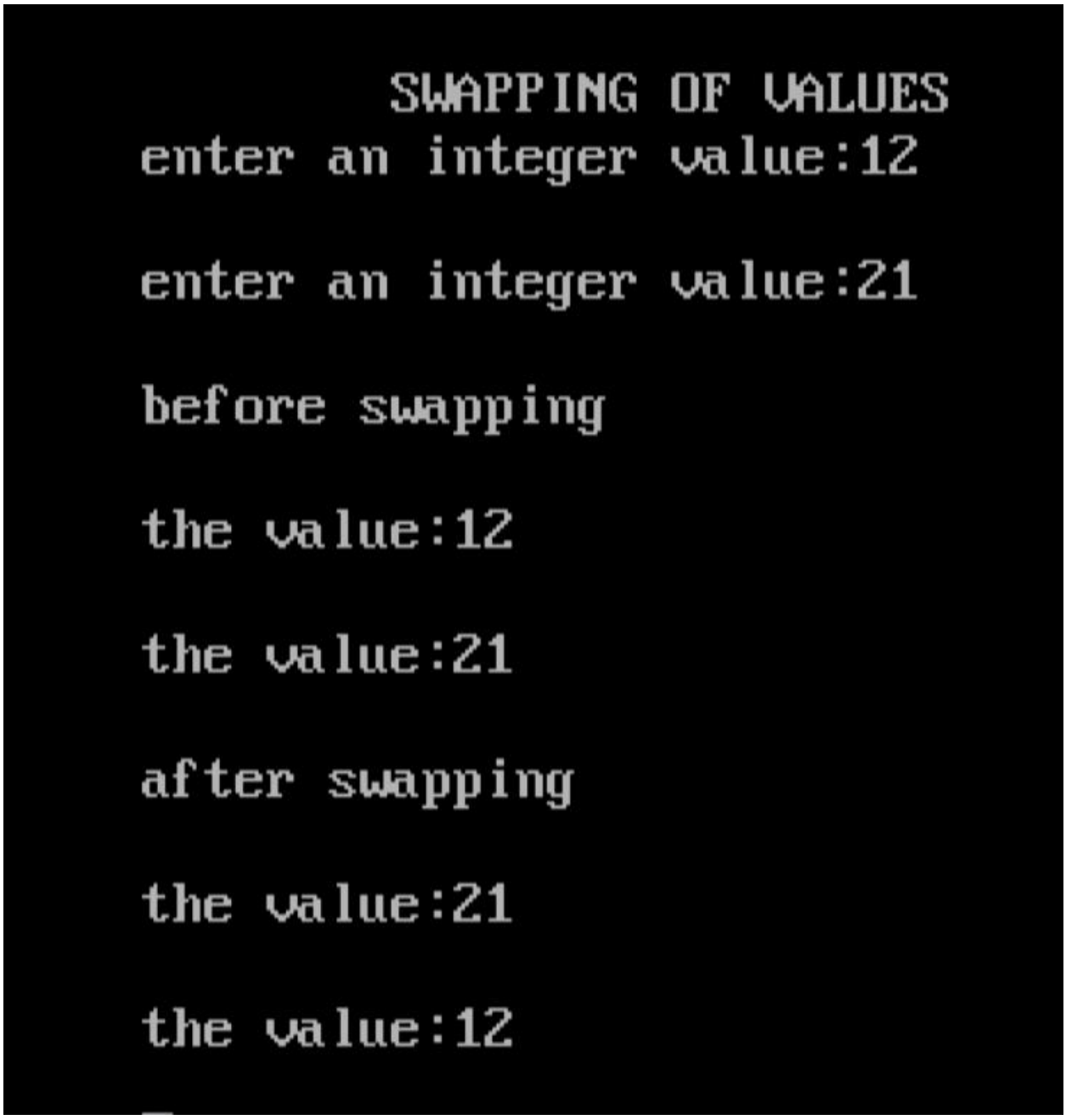
To write a c++ program to swap two values using friend function.

CODING:

```
#include<iostream.h>
#include<conio.h>
class sample
{
int a;
public:
void get(),put();
friend void exchange(sample &s1,sample &s2)
};
void exchange(sample &s1,sample &s2)
{
int t=s1.a;
s1.a=s2.a;
s2.a=t;
}
void sample::get()
{
cout<<"\nenter an integer value:";
cin>>a;
}
```

```
void sample::put()
{
cout<<"\nthe value:"<<a<<endl;
}
int main()
{
clrscr();
cout<<"\n\tSWAPPING OF VALUES";
sample x,y;
x.get();
y.get();
cout<<"\nbefore swapping\n";
x.put();
y.put();
exchange(x,y);
cout<<"\nafter swapping\n";
x.put();
y.put();
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus, the c++ program to perform friend function swapping of values is implemented and verified successfully.

Ex no:06 **CONSTRUCTOR AND DESTRUCTOR**

Date:

AIM:

To write a c++ program to implement constructor and destructor.

CODING:

```
#include<iostream.h>
#include<string.h>
#include<conio.h>
class str
{
char a[30];
public:
str(const char*s)
{
strcpy(a,s);
}
str(const char*s1,const char*s2)
{
strcpy(a,s1);
strcpy(a,s2);
}
void put()
{
cout<<"the string:"<<a<<endl;
}
~str()
```

```
{
cout<<"\nstring destroyed";
}
};
int main()
{
clrscr();
cout<<"\t\tCONSTRUCTOR AND DESTRUCTOR";
char p[30],q[30];
cout<<"\nenter two string:";
cin>>p>>q;
str a(p),b(p,q);
a.put();
b.put();
getch();
return 0;
}
```

OUTPUT:

RESULT:

Thus, the c++ program to perform constructor and destructor is implemented and verified successfully.

Ex no:07 **UNARY OPERATOR OVERLOADING**

Date:

AIM:

To write a c++ program to implement unary operator overloading.

CODING:

```
#include<iostream.h>
#include<conio.h>
class unary
{
int a;
float b;
public:
void getdata()
{
cout<<"\nenter an integer value:";
cin>>a;
cout<<"\nenter the float value:";
cin>>b;
}
void putdata()
{
cout<<"a="<<a<<"\t"<<"b="<<b<<endl;
}
void operator++()
{
++a;
```

```
++b;
}
void operator--()
{
--a;
--b;
}
void operator-()
{
a=-a;
b=-b;
}
};
int main()
{
clrscr();
cout<<"\n\t\tUNARY OPERATOR OVERLOADING";
unary u;
u.getdata();
cout<<"\npre-decreament operator overloading\n";
--u;
u.putdata();
cout<<"unary minus operator overloading\n";
-u;
u.putdata();
cout<<"pre-increament operator overloading\n";
++u;
u.putdata();
getch();
```

```
return 0;
}
```

OUTPUT:

```
                UNARY OPERATOR OVERLOADING
enter an integer value:3

enter the float value:3.4

pre-decreament operator overloading
a=2     b=2.4
unary minus operator overloading
a=-2    b=-2.4
pre-increament operator overloading
a=-1    b=-1.4
```

RESULT:

Thus,the c++program to perform unary function overloading is implemented and verified successfully.

Ex no:08 BINARY OPERATOR OVERLOADING

Date:

AIM:

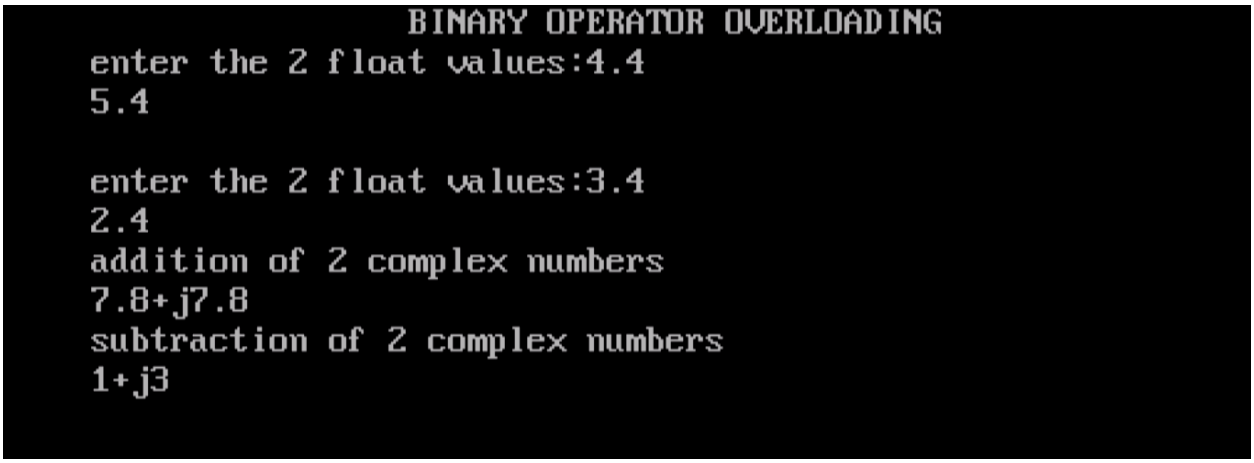
To write a c++ program to implement binary operator overloading.

CODING:

```
#include<iostream.h>
#include<conio.h>
class complex
{
float x,y;
public:
void getdata()
{
cout<<"\nenter the 2 float values:";
cin>>x>>y;
}
void putdata()
{
if(y>0)
cout<<x<<"+j"<<y<<endl;
else
{
y=-y;
cout<<x<<"-j"<<y<<endl;
}
}
complex operator+(complex);
friend complex operator-(complex,complex);
};
complex complex::operator+(complex c)
{
complex t;
t.x=x+c.x;
t.y=y+c.y;
return t;
}
complex operator-(complex t1,complex t2)
{
complex t3;
t3.x=t1.x-t2.x;
t3.y=t1.y-t2.y;
return t3;
}
```

```
int main()
{
clrscr();
cout<<"\t\tBINARY OPERATOR OVERLOADING";
complex c1,c2,c3;
c1.getdata();
c2.getdata();
c3=c1+c2;
cout<<"addition of 2 complex numbers\n";
c3.putdata();
c3=c1-c2;
cout<<"subtraction of 2 complex numbers\n";
c3.putdata();
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus, the c++ program to implement binary operator overloading is implemented and verified successfully.

Ex no:09 **HYBRID INHERITANCE**
Date:

AIM:

To write a c++ program to implement the hybrid inheritance.

CODING:

```
#include<iostream.h>
#include<conio.h>
class student
{
protected:
int rollnum;
public:
void getnum()
{
cout<<"\nenter the roll number:";
cin>>rollnum;
}
void putnum()
{
cout<<"\nroll no:"<<rollnum;
}
};
class test:public student
{
protected:
float mark1,mark2;
public:
void getmarks()
{
```

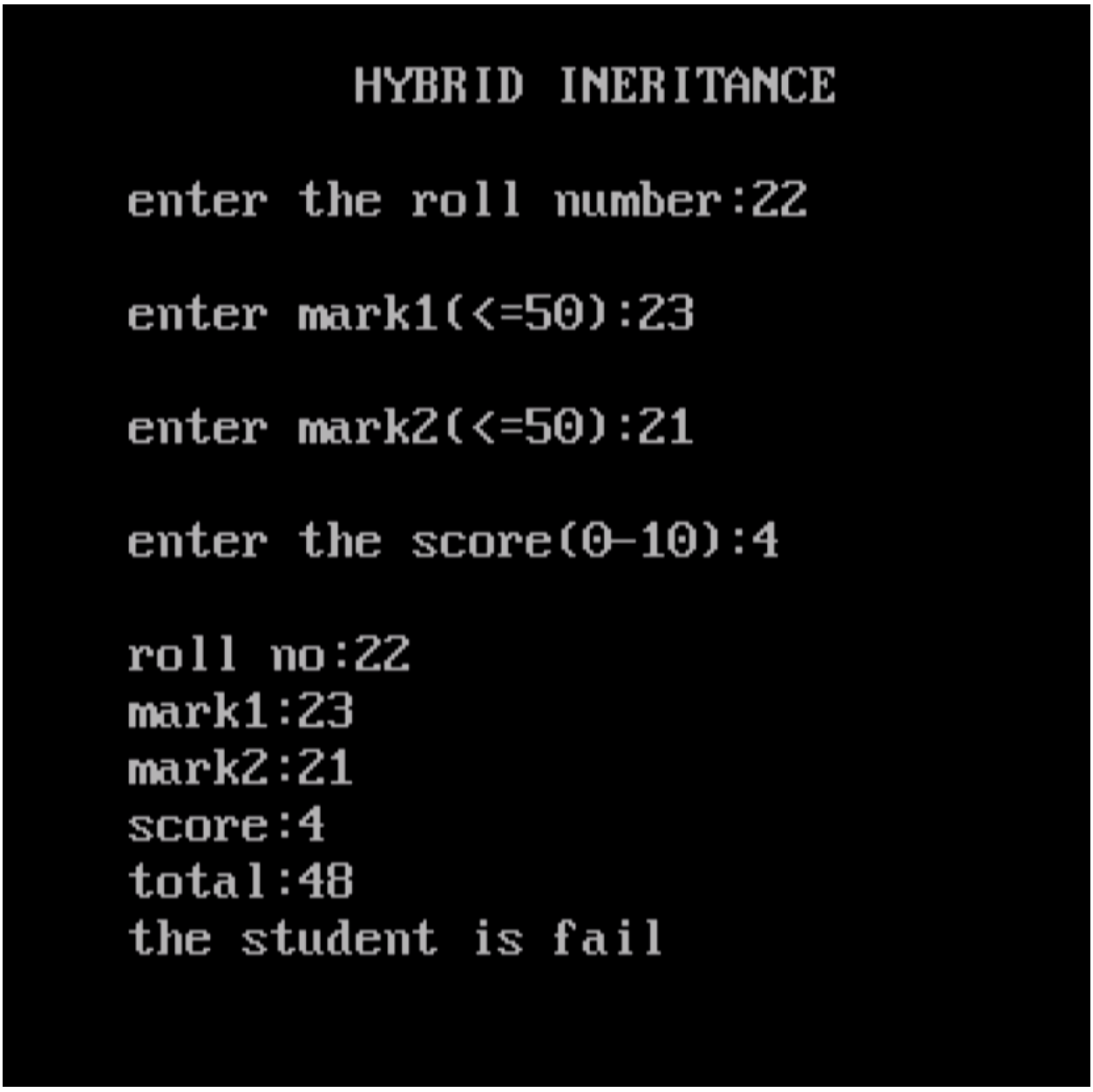
```
cout<<"\nenter mark1(<=50):";
cin>>mark1;
cout<<"\nenter mark2(<=50):";
cin>>mark2;
}
void putmarks()
{
cout<<"\nmark1:"<<mark1;
cout<<"\nmark2:"<<mark2;
}
};
class sports
{
protected:
float score;
public:
void getscore()
{
cout<<"\nenter the score(0-10):";
cin>>score;
}
void putscore()
{
cout<<"\nscore:"<<score;
}
};
class result:public test,public sports
{
float total;
```

```
public:
void display();
};
void result::display()
{
getnum();
getmarks();
getscore();
total=mark1+mark2+score;
putnum();
putmarks();
putscore();
cout<<"\ntotal:"<<total;
if(total<50)
cout<<"\nthe student is fail";
else
cout<<"\nthe student is pass";
}
int main()
{
clrscr();
cout<<"\n\tHYBRID INERITANCE\n";
result r;
r.display();
getch();
return 0;
}
```

OUTPUT 1:

```
        HYBRID INERITANCE

enter the roll number:12

enter mark1(<=50):35

enter mark2(<=50):45

enter the score(0-10):8

roll no:12
mark1:35
mark2:45
score:8
total:88
the student is pass
```

OUTPUT 2:



RESULT:

Thus, the c++ program to implement hybrid inheritance is implemented and verified successfully.

Ex no:10 **VIRTUAL FUNCTION**

Date:

AIM:

To write a c++ program using virtual function.

CODING:

```
#include<iostream.h>
#include<conio.h>
class media
{
protected:
char title[50];
float price;
public:
virtual void display()=0;
};
class book:public media
{
int pages;
public:
void getbook()
{
cout<<"\nenter the book name:";
cin>>title;
cout<<"\nenter the price of book:";
cin>>price;
cout<<"\nenter the no of pages in the books:";
```

```
cin>>pages;
}
void display();
};
class tape:public media
{
float time;
public:
void gettape()
{
cout<<"\nenter the tape name:";
cin>>title;
cout<<"\nenter the price of the tape:";
cin>>price;
cout<<"\nenter the playtime:";
cin>>time;
}
void display();
};
void book::display()
{
cout<<"\n\nbook title:"<<title;
cout<<"\ntotal pages:"<<pages;
cout<<"\nbook price:"<<price;
}
void tape::display()
{
cout<<"\n\ntape title:"<<title;
cout<<"\ntotal playtime:"<<time;
```

```
cout<<"\ntape price:"<<price;
}
int main()
{
clrscr();
cout<<"\nVIRTUAL FUNCTION";
book b;
b.getbook();
tape t1;
t1.gettape();
media*m;
m=&b;
m->display();
m=&t1;
m->display();
getch();
return 0;
}
```

OUTPUT:

```
VIRTUAL FUNCTION
enter the book name:harrypotter

enter the price of book:500

enter the no of pages in the books:155

enter the tape name:HISTORY

enter the price of the tape:600

enter the playtime:20


book title:harrypotter
total pages:155
book price:500

tape title:HISTORY
total playtime:20
tape price:600
```

RESULT:

Thus, the c++ program to Implement virtual function is implemented and verified successfully.

Ex no:11 **SIMPLE FILE OPERATION**

Date:

AIM:

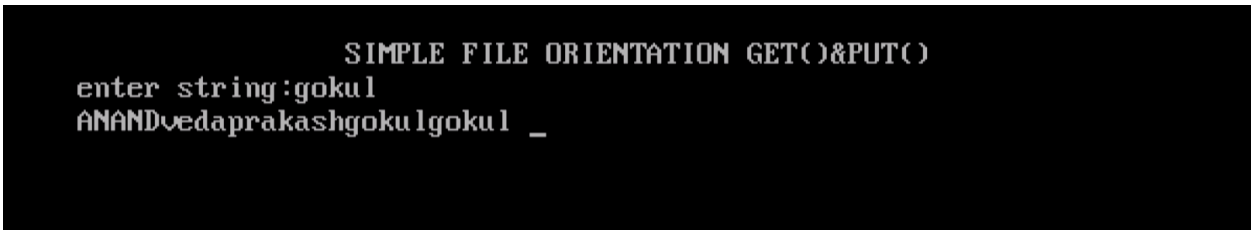
To write a c++ program for single file orientation using get() and put().

CODING:

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
#include<string.h>
int main()
{
clrscr();
cout<<"\n\t\tSIMPLE FILE ORIENTATION GET()&PUT()";
char str[80];
char ch;
cout<<"\nenter string:";
cin.getline(str,80);
int len=strlen(str);
fstream file;
file.open("ADDFILE",ios::in|ios::out|ios::app);
for(int i=0;i<len;i++)
{
file.put(str[i]);
}
file.seekg(0);
while(file)
```

```
{
file.get(ch);
cout<<(ch);
}
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus ,the c++ program for simple file operation using get() and put() is implemented and verified successfully.

DATA STRUCTURE

Ex no:01 IMPLEMENTATION OF STACK USING ARRAY

Date:

AIM:

To write a c++ program to implement stack using array data structure and perform a stack operation.

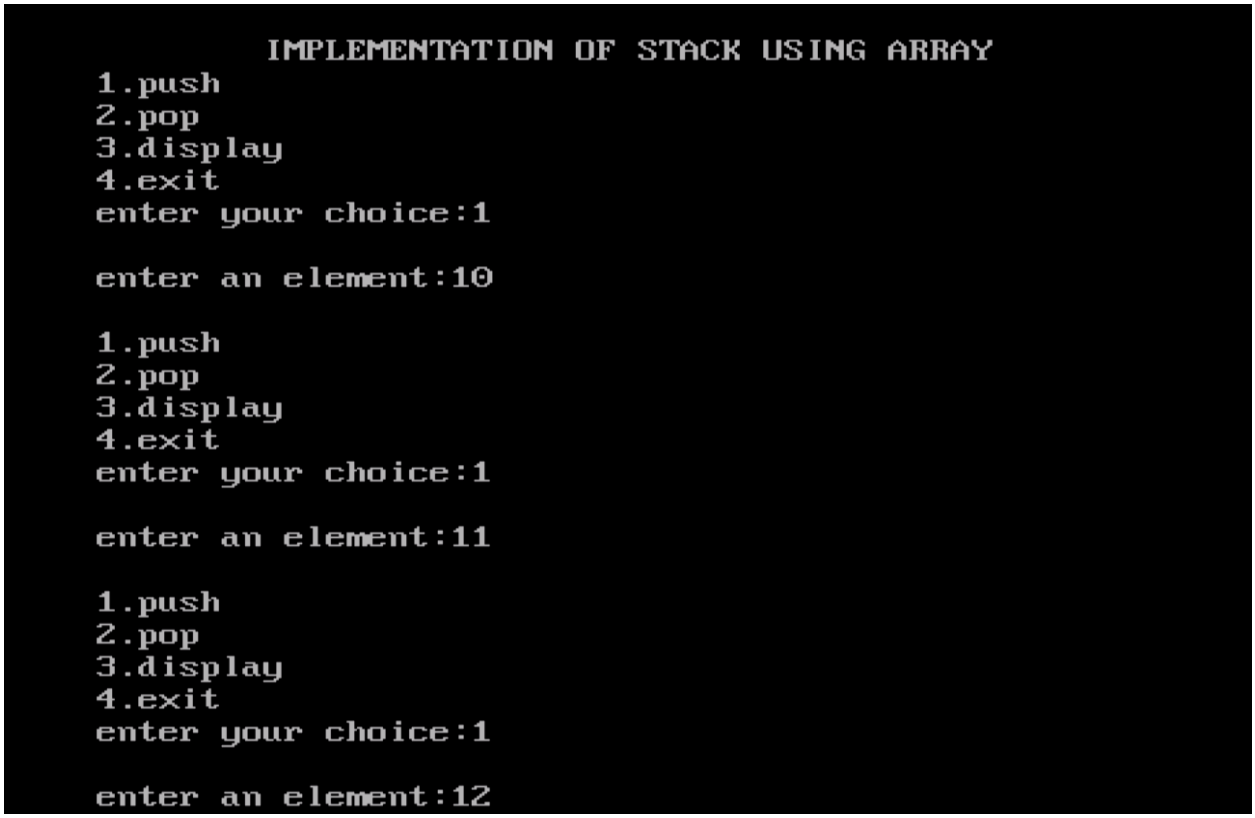
CODING:

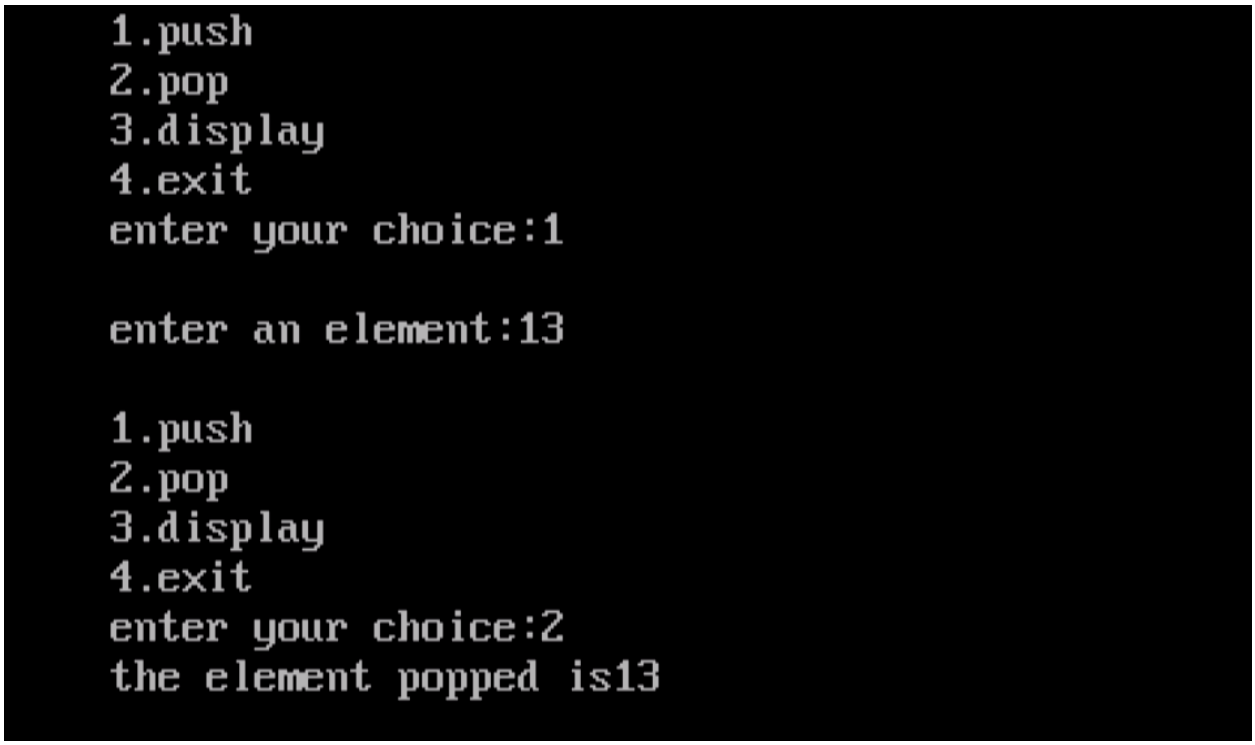
```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define max 5
Class stack
{
Int a[5],top;
Public:
Stack()
{
Top=-1;
}
Void push(),pop(),display();
};
Void stack::push()
{
Int x;
If(top==max-1)
cout<<"STACK OVERFLOW";
else
{
```

```
cout<<"\nenter an element:";
cin>>x;
top++;
a[top]=x;
}
}
Void stack::pop()
{
if(top==-1)
cout<<"STACK UNDERFLOW";
else
{
Cout<<"the element is popped is:"<<a[top]<<endl;
Top--;
}
}
Void stack::display()
{
If(top==-1)
Cout<<"STACK IS EMPTY";
Else
{
Cout<<"the elements are"<<endl;
For(int i=top;i>0;i--)
Cout<<a[i]<<endl;
}
}
Int main()
{
```

```
Clrscr();
Cout<<"\n\tIMPLEMENTATION OF STACK USING ARRAY";
STACK S;
Int choice;
While(1)
{
Cout<<"\n\n1.push\n2.pop\n3.display\n4.exit";
Cout<<"\nenter the choice:";
Cin>>choice;
Switch(choice)
{
Case 1:s.push();
break;
Case 2:s.pop();
break;
Case 3:s.display();
break;
Case 4:exit(0);
Default:cout<<"\n INVALID CHOICE ";
getch();
return 0;
}
}
}
```

OUTPUT 1:


IMPLEMENTATION OF STACK USING ARRAY
1.push
2.pop
3.display
4.exit
enter your choice:1
enter an element:10
1.push
2.pop
3.display
4.exit
enter your choice:1
enter an element:11
1.push
2.pop
3.display
4.exit
enter your choice:1
enter an element:12


1.push
2.pop
3.display
4.exit
enter your choice:1
enter an element:13
1.push
2.pop
3.display
4.exit
enter your choice:2
the element popped is13

```
1.push
2.pop
3.display
4.exit
enter your choice:3
the elements are
12
11

1.push
2.pop
3.display
4.exit
enter your choice:4
```

OUTPUT 2:

```
        IMPLEMENTATION OF STACK USING ARRAY
1.push
2.pop
3.display
4.exit
enter your choice:5

INVALID CHOICE
```

RESULT:

Thus, the c++ program to implement of stack using array is implemented and verified successfully.

Ex no:02 QUEUE IMPLEMENTATION OF ARRAY

Date:

AIM:

to write a c++ program to implement queue using array data structure and perform a queue operation.

CODING:

```
#include<iostream.h>
#include<conio.h>
int queue[100],n=100,front=-1,rear=-1;
void insert()
{
int val;
if(rear==n-1)
cout<<"queue overflow"<<endl;
else
{
if(front==-1)
front=0;
cout<<"insertion the element in queue:"<<endl;
cin>>val;
rear++;
queue[rear]=val;
}
}
void Delete()
{
if(front==-1||front>rear)
{
cout<<"queue underflow";
return;
}
else
{
cout<<"element deleted from queue is:"<<queue[front]<<endl;
front++;
}
}
void display()
{
if9front==-1)
cout<<"queue is empty"<<endl;
else
```

```
{
cout<<"queue elements are:";
for(int i=front;i<=rear;i++)
cout<<queue[i]<<" ";
cout<<endl;
}
}
int main()
{
int ch;
cout<<"1) insert element to queue"<<endl;
cout<<"2) delete element from queue"<<endl;
cout<<"3) display all the elements of queue"<<endl;
cout<<"4) exit"<<endl;
do
{
cout<<"\nenter your choice:"<<endl;
cin>>ch;
switch(ch)
{
case 1:insert();
break;
case 2:Delete();
break;
case 3:display();
break;
case 4:cout<<”exit”<<endl;
break;
default:cout<<"invalid choice"<<endl;
}
}
while(ch!=4);
return 0;
}
```

OUTPUT 1:

```
1) insert element to queue
2) delete element from queue
3) display all the elements of queue
4) exit

enter your choice:
1
insertion the element in queue:
11

enter your choice:
1
insertion the element in queue:
12

enter your choice:

1
insertion the element in queue:

13

enter your choice:
1
```

```
insertion the element in queue:
14

enter your choice:
3
queue elements are:11 12 13 14

enter your choice:
2
element deleted from queue is:11

enter your choice:
3
queue elements are:12 13 14

enter your choice:
4
```

OUTPUT 2:

```
1) insert element to queue
2) delete element from queue
3) display all the elements of queue
4) exit

enter your choice:
5
invalid choice
```

RESULT:

Thus ,the c++ program to implement queue using array data structure and to perform a queue operation is implemented and verified successfully

Ex no:03 **LINEAR SEARCH**

Date:

AIM:

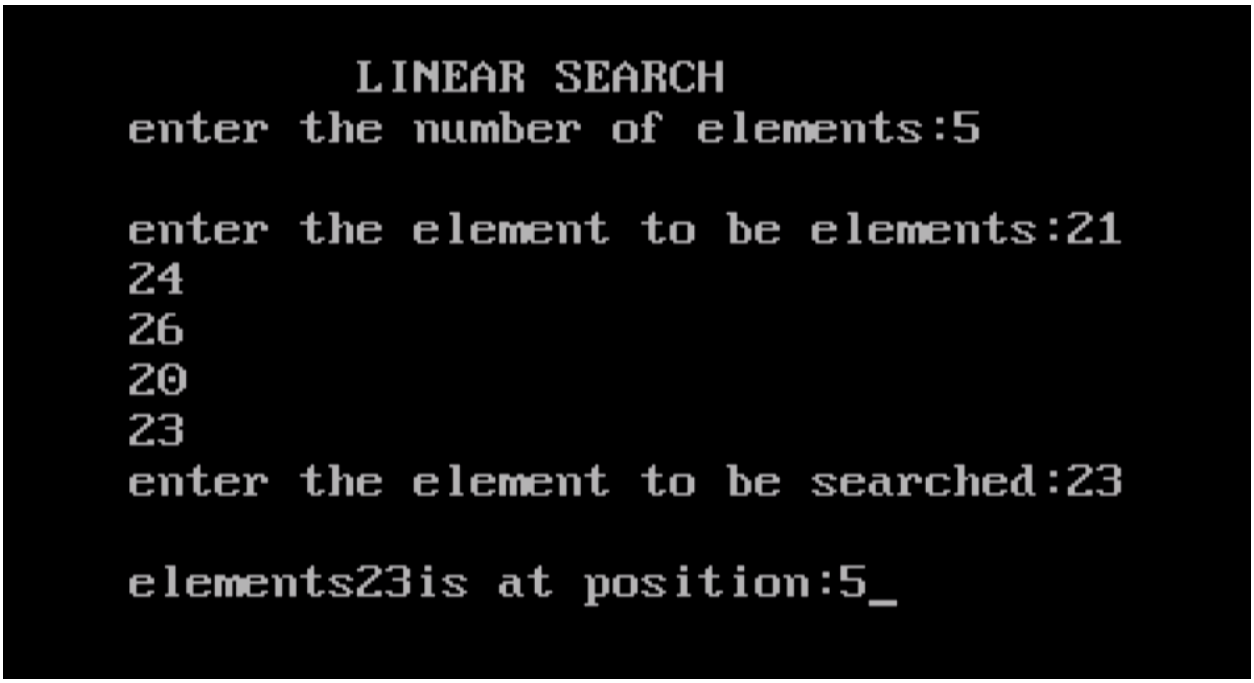
To write a c++ program to implement linear search.

CODING:

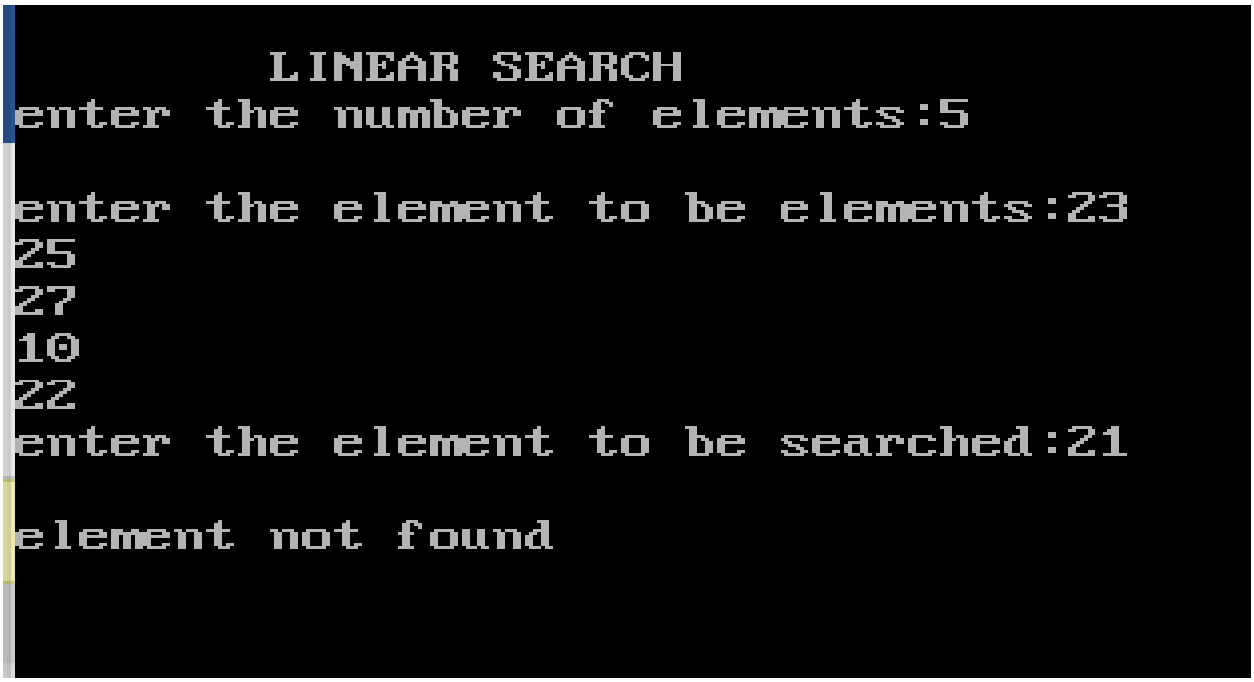
```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
class linear
{
int n,x,a[10],flag,pos;
public:
linear();
void find();
};
linear::linear()
{
cout<<"\nenter the number of elements:";
cin>>n;
cout<<"\nenter"<<n<<"element:";
for(int i=0;i<n;i++)
cin>>a[i];
cout<<"enter the element to be searched:";
cin>>x;
}
void linear::find()
{
```

```
flag=0;
for(int i=0;i<n;i++)
{
if(a[i]==x)
{
pos=i+1;
flag++;
}
}
if(flag>0)
cout<<"\nelements"<<x<<"is at position:"<<pos;
else
cout<<"\nelement not found";
}
int main()
{
clrscr();
cout<<"\n\tLINEAR SEARCH";
linear l;
l.find();
getch();
return 0;
}
```

OUTPUT 1:



OUTPUT 2:



RESULT:

Thus,the c++ program to implement linear search is implemented and verified successfully.

Ex no: 04 **BINARY SEARCH**

Date:

AIM:

To write a c++ program to implement binary search.

CODING:

```
#include<iostream.h>
#include<stdlib.h>
#include<conio.h>
class binary
{
int n,x,a[10],value,low,high,mid;
public:
binary();
void search();
};
binary::binary()
{
cout<<"\nnumber of elements:";
cin>>n;
cout<<"\nelements:";
for(int i=0;i<n;i++)
cin>>a[i];
cout<<"\nelement to be searched:";
cin>>x;
}
void binary::search()
{
low=0;high=n-1;
mid=(low+high)/2;
while((low<=high)&&(a[mid]!=x))
{
if(a[mid]<x)
low=mid+1;
else
high=mid-1;
mid=(low+high)/2;
}
if(a[mid]==x)
cout<<"\nelement"<<x<<"found at position"<<mid+1;
```

```
else
cout<<"\nelement not found";
}
void main()
{
clrscr();
cout<<"\nbinary search";
binary b;
b.search();
getch();
}
```

OUTPUT 1:

```
binary search
number of elements:5

elements:24
25
26
27
28

element to be searched:26

element26found at position3
```

OUTPUT 2:

```
binary search
number of elements:5

elements:24
25
26
27
10

element to be searched:
21

element not found_
```

RESULT:

Thus,the c++ program to implement binary seach is implemented and verified successfully.

Ex no:05 **BUBBLE SORT**

Date:

AIM:

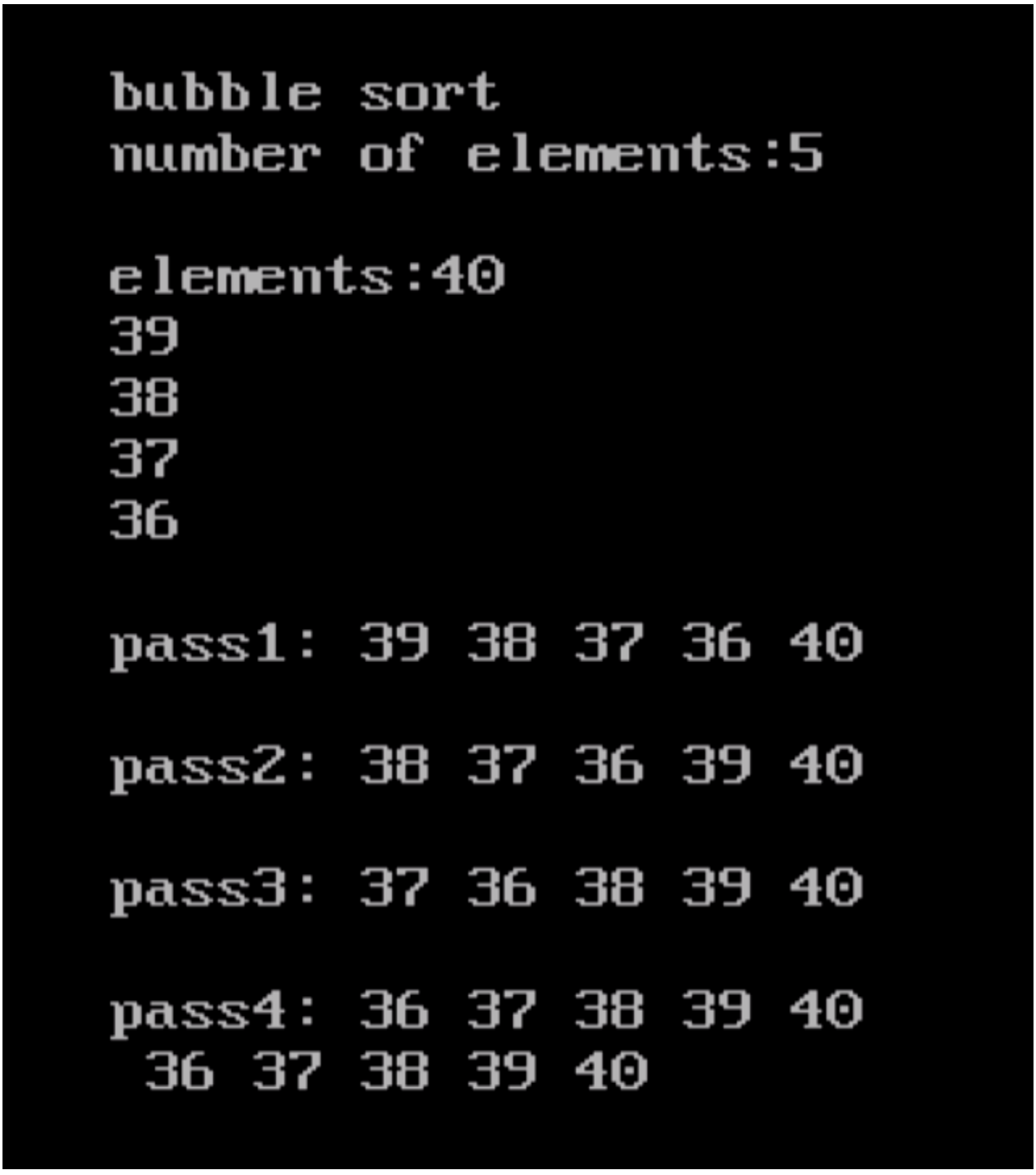
To write a c++ program to sort numbers using bubble sort.

CODING:

```
#include<iostream.h>
#include<conio.h>
class bubble
{
int a[10],n,t;
public:
bubble();
void sort();
void display();
};
bubble::bubble()
{
cout<<"\nnumber of elements:";
cin>>n;
cout<<"\nelements:";
for(int i=0;i<n;i++)
cin>>a[i];
}
void bubble::display()
{
for(int i=0;i<n;i++)
cout<<" "<<a[i];
cout<<endl;
}
void bubble::sort()
{
for(int i=0;i<n-1;i++)
{
for(int j=0;j<n-1;j++)
{
if(a[j]>a[j+1])
{
t=a[j];
a[j]=a[j+1];
a[j+1]=t;
```

```
}
}
cout<<"\npass"<<i+1<<":";
display();
}
}
void main()
{
clrscr();
cout<<"\nbubble sort";
bubble b;
b.sort();
b.display();
getch();
}
```

OUTPUT :



RESULT:

Thus,the c++ program to sort the numbers using bubble sort is implemented and verified successfully.

Ex no:06 **INSERTION SORT**

Date:

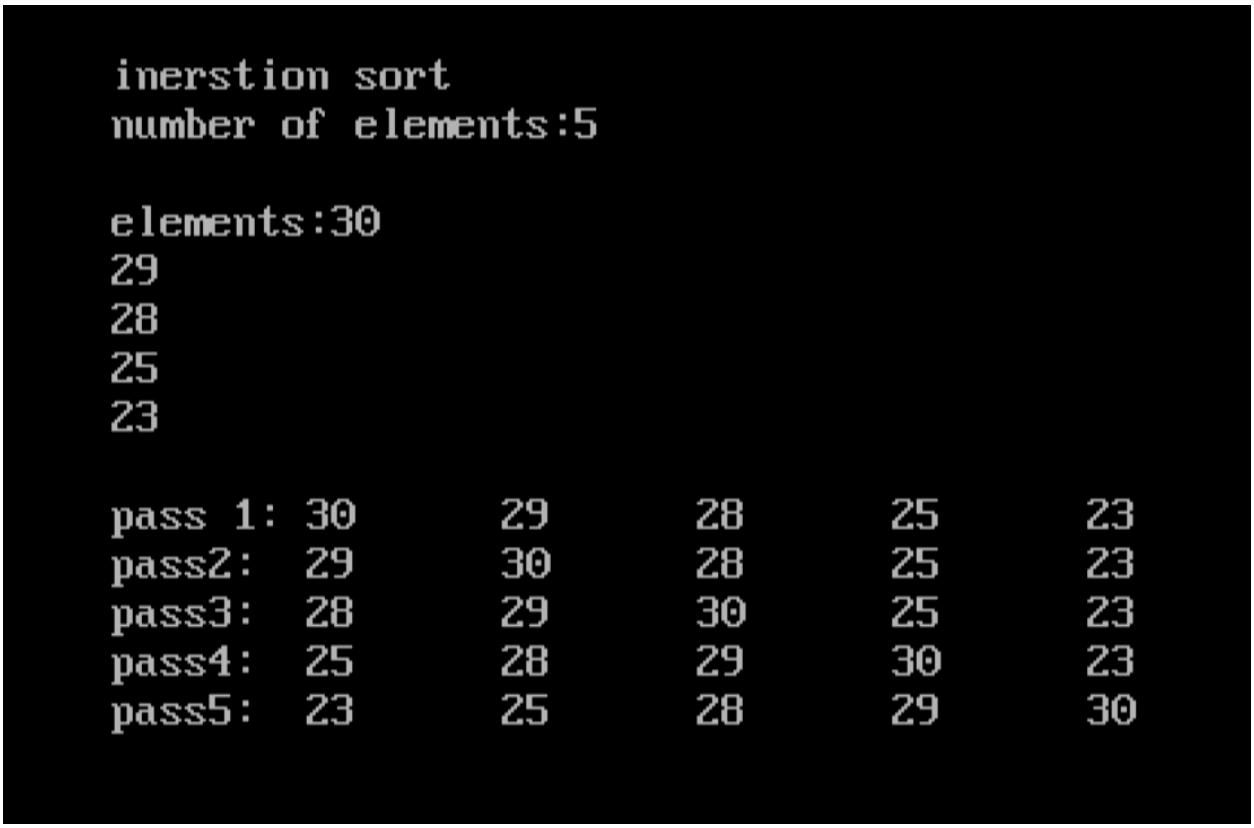
AIM:

To write a c++ program to sort numbers using insertion sort.

CODING:

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
cout<<"\ninsertion sort";
int n,a[50],i,pos,key;
cout<<"\nnumber of elements:";
cin>>n;
cout<<"\nelements:";
for(i=1;i<=n;i++)
cin>>a[i];
cout<<"\npass 1:";
for(int j=1;j<=n;j++)
{
key=a[i];pos=i;
while((pos>1)&&(a[pos-1]>=key))
{
a[pos]=a[pos-1];
pos=pos-1;
a[pos]=key;
}
cout<<"\npass"<<i<<":";
for(int j=1;j<=n;j++)
cout<<"\t"<<a[j];
}
getch();
}
```

OUTPUT:



RESULT:

thus, the c++ program to sort numbers using insertion sort is implemented and verified successfully.

Ex no:07 # SELECTION SORT

Date:

AIM:

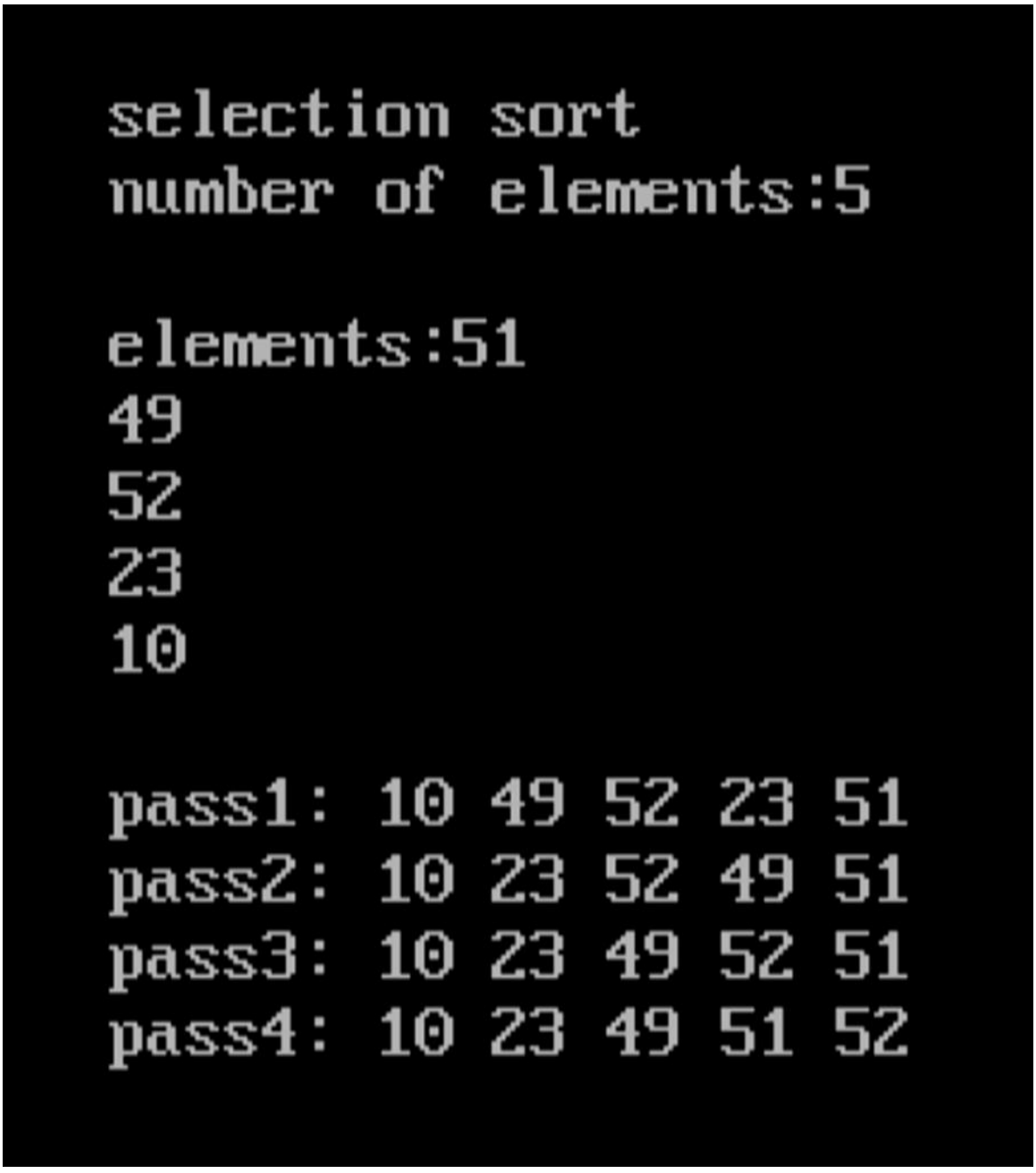
To write a c++ program to sort numbers using selection sort.

CODING:

```
#include<iostream.h>
#include<conio.h>
class selection
{
int a[10],n;
public:
selection();
void sort();
};
selection::selection()
{
cout<<"\nnumber of elements:";
cin>>n;
cout<<"\nelements:";
for(int i=0;i<n;i++)
cin>>a[i];}
void selection::sort()
{
int key,pos,i,j,k,t;
for(i=0;i<n-1;i++)
{
key=a[i];pos=i;
for(j=i+1;j<n;j++)
if(key>a[j])
{key=a[j];pos=j;}
t=a[i];
a[i]=key;
a[pos]=t;
cout<<"\npass"<<i+1<<":";
for(k=0;k<n;k++)
cout<<" "<<a[k];
}
}
void main()
{
clrscr();
cout<<"\nselection sort";
```

```
selection s;
s.sort();
getch();
}
```

OUTPUT:



RESULT:

Thus, the c++ program to sort numbers using selection sort is verified successfully.

Ex no:08 **TREE TRAVERSAL INORDER**
Date:

AIM:

To write a c++ program to create a binary tree and perform inorder traversal to visit all the nodes of it.

CODING:

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int data;
struct node*left,*right;
};
struct node*root=NULL;
struct node*create(int data)
{
struct node*newnode;
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=data;
newnode->left=NULL;
newnode->right=NULL;
return(newnode);
}
void insertion(struct node**node,int data)
{
if(!*node)
*node=create(data);
else if(data<=(*node)->data)
insertion(&(*node)->left,data);
else if(data>=(*node)->data)
insertion(&(*node)->right,data);
}
void inorder(struct node*node)
{
if(node)
{
inorder(node->left);
cout<<node->data<<" ";
inorder(node->right);
}
}
int main()
```

```
{
clrscr();
cout<<"\n\t\tTREE TRAVERSAL INORDER";
int data,n;
cout<<"\nHOW MANY NODES?\n";
cin>>n;
for(int i=1;i<=n;i++)
{
cout<<"\nenter the data:";
cin>>data;
insertion(&root,data);
}
cout<<"\nthe inorder traversal is:";
inorder(root);
return 0;
}
```

OUTPUT:

```
                TREE TRAVERSAL INORDER
HOW MANY NODES?
5

enter the data:17

enter the data:25

enter the data:11

enter the data:10

enter the data:29

the inorder traversal is:10 11 17 25 29
```

RESULT:

Thus, the c++ program to create a binary tree and perform inorder traversal to visit all the nodes of it is implemented and verified successfully.

Ex no:09 **TREE TRAVERSAL PREORDER**

Date:

AIM:

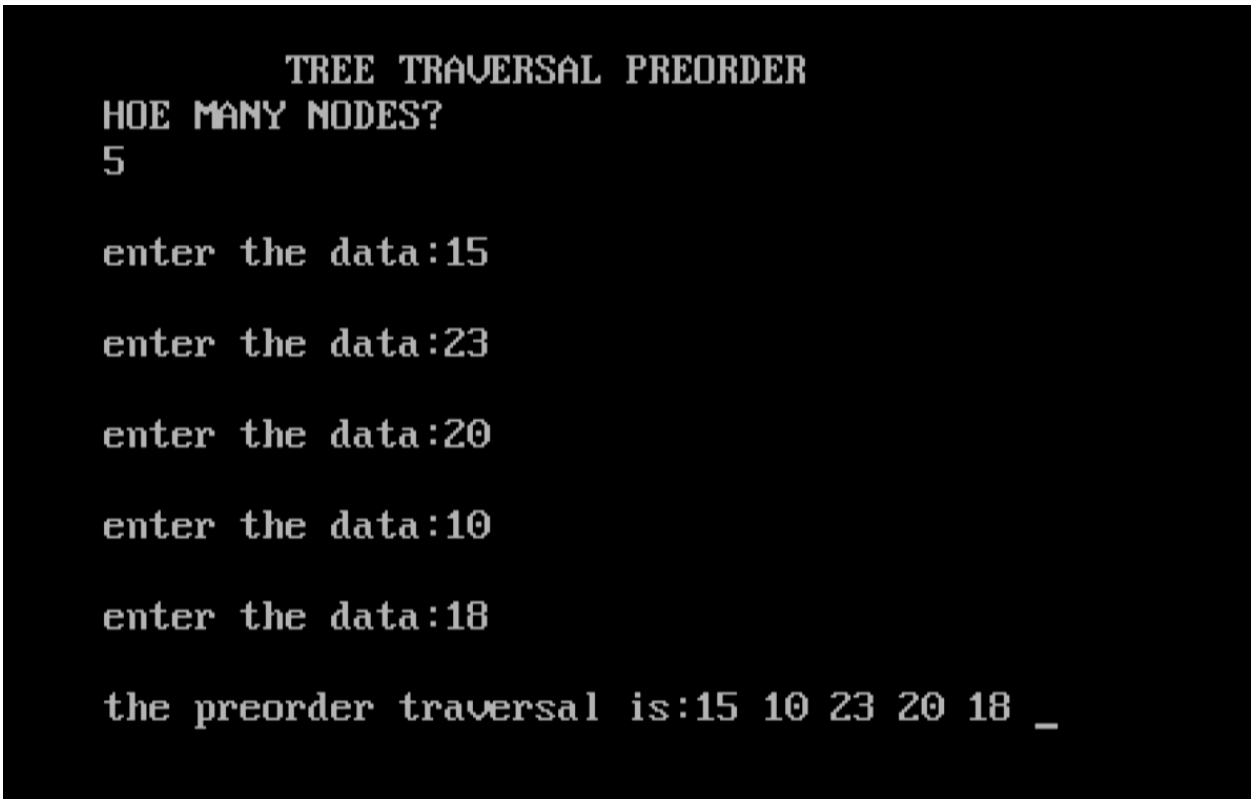
To write a c++ program to create a binary tree and perform preorder traversal to visit all the nodes of it

CODING:

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int data;
struct node*left,*right;
};
struct node*root=NULL;
struct node*create(int data)
{
struct node*newnode;
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=data;
newnode->left=NULL;
newnode->right=NULL;
return(newnode);
}
void insertion(struct node**node,int data)
{
if(!*node)
*node=create(data);
else if(data<(*node)->data)
insertion(&(*node)->left,data);
else if(data>(*node)->data)
insertion(&(*node)->right,data);
}
void preorder(struct node*node)
{
if(node)
{
cout<<node->data<<" ";
preorder(node->left);
preorder(node->right);
}
}
```

```
int main()
{
clrscr();
cout<<"\n\tTREE TRAVERSAL PREORDER";
int data,n;
cout<<"\nHOE MANY NODES?\n";
cin>>n;
for(int i=1;i<=n;i++)
{
cout<<"\nenter the data:";
cin>>data;
insertion(&root,data);
}
cout<<"\nthe preorder traversal is:";
preorder(root);
return 0;
}
```

OUTPUT:



RESULT:

Thus ,the c++ program to create a binary tree and perform preorder to visit all the nodes of it is implemented and verified successfully.

Ex no:10 **TREE TRAVERSAL POSTORDER**

Date:

AIM:

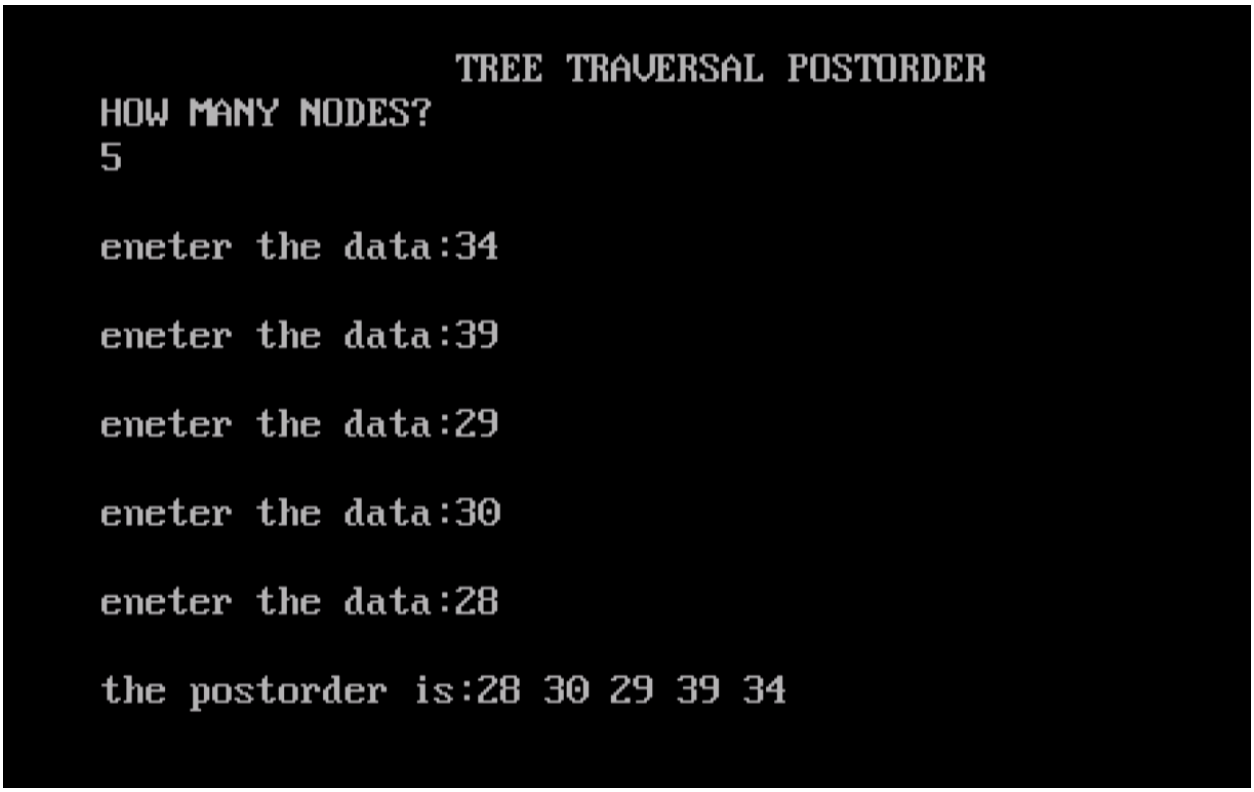
To writ a c++ program to create a binary tree and perform postorder traversal to visit all the nodes of it.

CODING:

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int data;
struct node*left,*right;
};
struct node*root=NULL;
struct node*create(int data)
{
struct node*newnode;
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=data;
newnode->left=NULL;
newnode->right=NULL;
return(newnode);
}
void insertion(struct node**node,int data)
{
if(!*node)
*node=create(data);
else if(data<(*node)->data)
insertion(&(*node)->left,data);
else if(data>(*node)->data)
insertion(&(*node)->right,data);
}
void postorder(struct node*node)
{
if(node)
{
postorder(node->left);
postorder(node->right);
cout<<node->data<<" ";
}
}
```

```
int main()
{
clrscr();
cout<<"\n\t\tTREE TRAVERSAL POSTORDER";
int data,n;
cout<<"\nHOW MANY NODES?\n";
cin>>n;
for(int i=1;i<=n;i++)
{
cout<<"\neneter the data:";
cin>>data;
insertion(&root,data);
}
cout<<"\nthe postorder is:";
postorder(root);
return 0;
}
```

OUTPUT:



RESULT:

Thus, the c++ program to create a binary tree and perform postorder traversal to visit all the nodes of it is implemented and verified successfully.

Ex.no:11 **SINGLY LINKED LIST**

Date:

AIM:

to write a c++ program to implement singly linked list.

CODING:

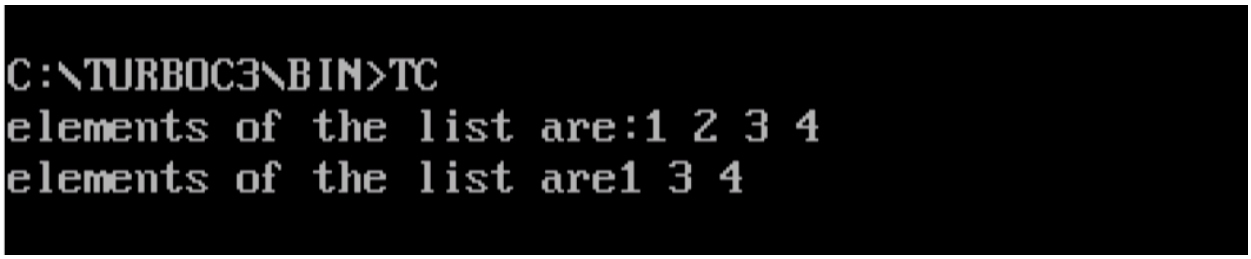
```
#include<iostream.h>
#include<conio.h>
class node
{
public:
int data;
node*next;
node()
{
data=0;
next=NULL;
}
node(int data)
{
this->data=data;
this->next=NULL;
}
};
class linkedlist
{
node*head;
```

```
public:
linkedlist()
{
head=NULL;
}
void insertnode(int);
void printlist();
void deletenode(int);
};
void linkedlist::deletenode(int nodeoffset)
{
node*temp1=head,*temp2=NULL;
int listlen=0;
if(head==NULL)
{
cout<<"list empty"<<endl;
return;
}
while(temp1!=NULL)
{
temp1=temp1->next;
listlen++;
}
if(listlen<nodeoffset)
{
cout<<"index out of range"<<endl;
return;
}
temp1=head;
```

```
if(nodeoffset==1)
{
head=head->next;
delete temp1;
return;
}
while(nodeoffset-->1)
{
temp2=temp1;
temp1=temp1->next;
}
temp2->next=temp1->next;
delete temp1;
}
void linkedlist::insertnode(int data)
{
node*newnode=new node(data);
if(head==NULL)
{
head=newnode;
return;
}
node*temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=newnode;
}
```

```
void linkedlist::printlist()
{
node*temp=head;
if(head==NULL)
{
cout<<"list empty"<<endl;
return;}
while(temp!=NULL)
{
cout<<temp->data<<" ";
temp=temp->next;
}
}
int main()
{
linkedlist list;
list.insertnode(1);
list.insertnode(2);
list.insertnode(3);
list.insertnode(4);
cout<<"elements of the list are:";
list.printlist();
cout<<endl;
list.deletenode(2);
cout<<"elements of the list are";
list.printlist();
cout<<endl;
return 0;
}
```

OUTPUT:



RESULT:

Thus, the c++ program to implement singly linked list is implemented and verified successfully.