SIR C R R COLLEGE OF ENGINEERING, ELURU

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB MANUAL FOR A.Y: 2016-17

SUBJECT	:	DATA STRUCTURES
YEAR	:	2016-17
SEMESTER	:	2/4 CSE, I-SEM

Faculty Members:

1. M.Madhava Rao
2. B H Benny
3. Samparthi. V.S. Kumar

INDEX

LIST OF ADD-ON PROGRAMMES

S.NO	PROGRAMME NAME
1	**Write a program to find the Factorial of a given number.**
2	**Write a program to find GCD of two given numbers.**
3	**Write a program for Towers of Hanoi Problem.**
4	**Write a program to convert Infix to prefix expression.**
5	**Write a program for operations on Queue.**
6	**Write a program for operations on Single linked list.**
7	**Write a program for operations on Double linked list.**
8	**Write a program insertion sort implementation using array s in ascending order.**
9	**Write a program Selection sort implementation using array ascending.**
10	**Write a program for sorting array using shell sorting method.**
11	**Write a program to Traverse the Tree Non-Recursively**.
	Write a program to Implement Priority Queue to Add and Delete Elements.

/*Write a program for stack operations.*/

```
#include<stdio.h>
#include<stdlib.h>
#define stack_size 5
int item,s[5],top=-1,item_deleted;
void push();
int pop();
void display();
main()
{
int choice;
for( ; ; )
{
printf("enter choice");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter item");
       scanf("%d",&item);
       push();
       break;
case 2:printf("delete element from top of stack\n");
       item_deleted=pop();
       printf("deleted item is %d",item_deleted);
       break;
case 3:printf("display elements\n");
       display();
       break;
default:printf("invalid choice");
        exit(0);
}
}
}
void push()
{
if(top==stack_size-1)
{
printf("stack overflow\n");
return;
}
top=top+1;
s[top]=item;
}
```

```
int pop()
{
if(top==-1)
{
return 0;
}
item_deleted=s[top--];
return item_deleted;
}
void display()
{
int i;
if(top==-1)
{
printf("stack is empty");
return;
}
printf("contents of stack");
for(i=0;i<=top;i++)
{
printf("%d\t",s[i]);
}
}
```

RESULT APPEARED ON SCREEN (INPUT AND OUTPUT):-

```
enter choice1
enter item4
enter choice1
enter item5
enter choice1
enter item6
enter choice1
enter item7
enter choice1
enter item8
enter choice1
enter item9
stack overflow
enter choice2
delete element from top of stack
deleted item is 8
enter choice3
display elements
contents of stack4       5          6          7
enter choice2
delete element from top of stack
deleted item is 7
enter choice2
delete element from top of stack
deleted item is 6
enter choice2
delete element from top of stack
deleted item is 5
enter choice2
delete element from top of stack
deleted item is 4
enter choice3
display elements
stack is empty
enter choice4
invalid choice
```

/* Write a program to convert Infix to postfix expression.*/

```
#include<stdio.h>
#include<string.h>
int f(char symbol)
{
switch(symbol)
{
case '+':
case '-':return 2;
case '*':
case '/':
case '%':return 4;
case '$':
case '^':return 5;
case '(':return 0;
case '#':return -1;
default:return 8;
}
}
int g(char symbol)
{
switch(symbol)
{
case '+':
case '-':return 1;
case '*':
case '/':
case '%':return 3;
case '$':
case '^':return 6;
case '(':return 9;
case ')':return 0;
default:return 7;
}
}
void infix_postfix(char infix[],char postfix[])
{
int i=0,j=0,top=-1;
char symbol,s[30];
s[++top]='#';
for(i=0;i<strlen(infix);i++)
{
symbol=infix[i];
while(f(s[top])>g(symbol))
{
postfix[j]=s[top--];
j++;
}
if(f(s[top])!=g(symbol))
{
s[++top]=symbol;
}
else
{
top--;
}
}
while(s[top]!='#')
{
postfix[j++]=s[top--];
}

postfix[j]='\0';
}
main()
{
char infix[40],postfix[40];
printf("enter infix expression\n");
scanf("%s",infix);
infix_postfix(infix,postfix);
printf("postfix expression is\n");
printf("%s",postfix);
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter the infix expression

(a+b)*c-d


 The postfix expression is ab+c*d-
```

/*Write a program to Evaluate postfix expression.*/

```
#include<stdio.h>
#include<string.h>
#include<math.h>
double compute(char symbol,double op1,double op2)
{
int i;
double k=1;
switch(symbol)
{
case '+':return op1+op2;
case '-':return op1-op2;
case '*':return op1*op2;
case '/':return op1/op2;
case '$':
case '^':for(i=1;i<=op2;i++)
        k=k*op1;
        return k;
}
}
main()
{
double s[20],op1,op2,res;
int i,top=-1;
char postfix[20],symbol;
printf("enter post fix expression\n");
scanf("%s",postfix);
for(i=0;i<strlen(postfix);i++)
{
symbol=postfix[i];
if(isdigit(symbol))
s[++top]=symbol-'0';
else
{
op2=s[top--];
op1=s[top--];
res=compute(symbol,op1,op2);
s[++top]=res;
}
}
res=s[top--];
printf("the result is%f",res);
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter the postfix expression
84-73+*

The value of postfix expression is 40.000000
```

/*Write a program for implemt the Towers of Hanoi problem.*/

```
##include<stdio.h>
#include<math.h>
int tower(int,char,char,char);
main()
{
int n;
char source,temp,destination;
printf("enter no.of discs\n");
scanf("%d",&n);
source='a';
temp='b';
destination='c';
tower(n,source,temp,destination);
}
int tower(int n,char source,char temp,char destination)
{
if(n==1)
{
printf("move %d disc from %c to %c\n",n,source,destination);
return;
}
tower(n-1,source,destination,temp);
printf("move %d disc from %c to %c\n",n,source,destination);
tower(n-1,temp,source,destination);
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter the number of disks

3

move the 1 disc from a to c

move the 2 disc from a to b

move the 1 disc from c to b

move the 3 disc from a to c

move the 1 disc from b to a

move the 2 disc from b to c

move the 1 disc from a to c
```

/* Write a program for converting given infix expression to prefix expression.*/

```
#include<stdio.h>
#include<string.h>
int f(char symbol)
{
switch(symbol)
{
case '+':
case '-':return 1;
case '*':
case '%':
case '/':return 3;
case '$':
case '^':return 6;
case ')':return 0;
case '#':return -1;
default :return 8;
}
}
int g(char symbol)
{
switch(symbol)
{
case '+':
case '-':return 2;
case '*':
case '%':
case '/':return 4;
case '$':
case '^':return 5;
case '(':return 0;
case ')':return 9;
default :return 7;
}
}
void infix_prefix(char infix[],char prefix[])
{
int i=0,j=0,top=-1;
char temp;
char symbol,s[20];
s[++top]='#';
for(i=0,j=strlen(infix)-1;i<j;i++,j--)
{
temp=infix[i];
infix[i]=infix[j];
infix[j]=temp;
}
j=0;
for(i=0;i<strlen(infix);i++)
{
symbol=infix[i];
while(f(s[top])>g(symbol))
{
prefix[j]=s[top--];
j++;
}
if(f(s[top])!=g(symbol))
s[++top]=symbol;
else
top--;
}
while(s[top]!='#')
prefix[j++]=s[top--];
prefix[j]='\0';
for(i=0,j=strlen(prefix)-1;i<j;i++,j--)
{
temp=prefix[i];
prefix[i]=prefix[j];
prefix[j]=temp;
}
j=0;
}
main()
{
char infix[40],prefix[40];
printf("enter the infix expression\n");
scanf("%s",infix);
infix_prefix(infix,prefix);
printf("the prefix expression is%s",prefix);
}
```

RESULT APPERED ON SCREEN (I/P & O/P)

```
Enter the infix expression

(a+b)*c+d


The prefix expression is +*+abcd
```

/* Write a program to find the Factorial of a given number */

```
#include<stdio.h>
int fact(int n)
{
if(n==0)
return 1;
else
return n*fact(n-1);
}
main()
{
int n,k;
printf("Enter any number\n");
scanf("%d",&n);
k=fact(n);
printf("the factorial of %d is=%d",n,k);
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter any nunber
6
the factorial of 6 is=720
```

/* Write a program to find GCD of two given numbers */

```
#include<stdio.h>
int gcd(int m,int n)
{
if(n==0) return m;
else if(n>m) return gcd(n,m);
else if(m>n) return gcd(n,m%n);
}
main()
{
int m,n,k;
printf("Enter any two numbers\n");
scanf("%d%d",&m,&n);
k=gcd(m,n);
printf("The GCD( %d, %d)=%d",m,n,k);
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter any two numbers
18
24

The GCD(18,24)=6
```

/* Write a program for operations on Queue.*/

```
#include<stdio.h>
#define queue_size 5
int item,q[5],f=0,r=-1;
void insert_rear()
{
if(r==queue_size-1)
{
printf("queue is over flow");
return;
}
r=r+1;
q[r]=item;
}
void delete_front()
{
if(f>r)
{
printf("queue is under flow");
return;
}
printf("deleted item is%d",q[f++]);
if(f>r)
{
f=0;
r=-1;
}
}
void display()
{
if(f>r)
{
printf("queue is empty");
return;
}
printf("contents of queue:\n");
int t,i;
t=f;
for(i=f;i<=queue_size-1;i++)
{
printf("%d\t",q[t++]);
}
}
main()
```

```
{
int ch;
for( ; ; )
{
printf("entert the choice");

scanf("%d",&ch);
switch(ch)

{
case 1 :printf("enter item:");
        scanf("%d",&item);
        insert_rear();
        break;
case 2 :delete_front();
        break;
case 3 :display();
        break;
default :exit(0);
}
}
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
1.insert 2.delete 3.display 4.exit
Enter your choice
1
Enter element to be inserted
12
 1.insert 2.delete 3.display 4.exit
Enter your choice
1
Enter element to be inserted
34

 1.insert 2.delete 3.display 4.exit


Enter your choice
3
The elements in the Queue are 12     34
 1.insert 2.delete 3.display 4.exit
Enter your choice
2
the deleted element is 12
 1.insert 2.delete 3.display 4.exit
Enter your choice
4
```

/* Write a program for operations on Circular Queue.*/

```
#include<stdio.h>
#define queue_size 5
int item,count=0,f=0,q[5],r=-1,ch;
void insert_rear()
{
if(count==queue_size)
{
printf("queue is overflow");
return;
}
else
{
r=(r+1)%queue_size;
q[r]=item;
count++;
}
}
void delete_front()
{
if(count==0)
{
printf("queue underflow");
return;
}
else
{
printf("deleted item is %d",q[f]);
f=(f+1)%queue_size;
count--;
}
}
void display()
{
if(count==0)
{
printf("queue is empty");
return;
}
else
{

   printf("contents of queue\n");
int i,t;
t=f;
for(i=1;i<=count;i++)
{
printf("%d\t",q[t]);
t=(t+1)%queue_size;
}
}
}
main()
{
for( ; ; )
{
printf("enter choice");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("enter item");
        scanf("%d",&item);
        insert_rear();
        break;
case 2:delete_front();
        break;
case 3:display();
        break;
default :exit(0);
}
}
}
```

RESULT APPERED ON SCREEN (I/P & O/P)

```
1.insert 2.delete 3.display 4.exit
Enter your choice
1
Enter element
12

1.insert 2.delete 3.display 4.exit
Enter your choice
1
Enter element
34

1.insert 2.delete 3.display 4.exit

Enter your choice

1

Enter element

67


1.insert 2.delete 3.display 4.exit
Enter your choice
1
Enter element
66
over flow
1.insert 2.delete 3.display 4.exit
Enter your choice
2
The deleted element is  12
1.insert 2.delete 3.display 4.exit
Enter your choice
1
Enter element
45
1.insert 2.delete 3.display 4.exit
Enter your choice
3
   34   67   45
1.insert 2.delete 3.display 4.exit
Enter your choice
4
```

/*Write a program for implementing the operations of dequeue. */

```
#include<stdio.h>
#define queue_size 5
int item,f=0,r=-1,ch,q[5];
void insert_rear()
{
if(r==queue_size-1)
{
printf("queue is overflow");
return;
}
r=r+1;
q[r]=item;
}
void delete_front()
{
if(f>r)
{
printf("queue underflow");
return;
}
printf("deleted item is %d",q[f++]);
if(f>r)
{
f=0;
r=-1;
}
}
void insert_front()
{
if(f==0&&r==-1)
{
q[++r]=item;
return;
}
if(f!=0)
{
q[--f]=item;
return;
}
printf("front insertion not possible");
}
void delete_rear()
{
if(f>r)
```

```
{
printf("queue is underflow");
return;
}
printf("deleted item is %d",q[r--]);
if(f>r)
{
f=0;
r=-1;
}
}
void display()
{
int i,t;
t=f;
if(f>r)
{
printf("queue is empty");
return;
}
printf("contents of queue");
for(i=f;i<=r;i++)
{
printf("%d\t",q[t++]);
}
}
main()
{
for( ; ; )
{
printf("enter choice");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("enter item");
       scanf("%d",&item);
       insert_rear();
       break;
case 2:delete_front();
       break;
case 3:printf("enter item");
       scanf("%d",&item);
       insert_front();
       break;
case 4:delete_rear();
       break;
case 5:display();
       break;
default :exit(0);
}
}
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
Enter your choice
1
Enter element to be insert
12

1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
Enter your choice
1
Enter element to be insert
34

1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
Enter your choice
1
Enter element to be insert
67

1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
Enter your choice


5
The elements in the queue are  12     34       67

1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
Enter your choice
1
Enter element to be insert
11
over flow

1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
```

```
Enter your choice
3
The deleted element is 12

1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
Enter your choice
4
The deleted element is 67

1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
Enter your choice
5
The elements in the queue are  34

1.insert_rear 2.insert_front 3.delete_front 4.delete_rear 5.display 6.exit
Enter your choice
6
```

/* Write a program to implement the operations on Single Linked List. */

```
#include<stdio.h>
#include<stdlib.h>
int item,ch;
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE first;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("out of memory\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(int item,NODE first)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=first;
return temp;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("empty list");
return;
}
temp=first;
```

```
printf("item deleted is%d",temp->info);
temp=temp->link;
freenode(first);
return temp;
}
NODE insert_rear(int item,NODE first)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
{
printf("list empty");
return temp;
}
cur=first;
while(cur->link!=NULL)


                                    {
cur=cur->link;
}
cur->link=temp;
return first;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list empty");
return first;
}
if(first->link==NULL)
{
printf("the item to be deleted %d\n",first->info);
freenode(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
```

```
}
prev->link=NULL;
printf("item deleted is%d",cur->info);
freenode(cur);
return first;
}
void display()
{
if(first==NULL)
{
printf("list empty");
return;
}
NODE temp;
temp=first;
while(temp!=NULL)
{
printf("%d\t",temp->info);
temp=temp->link;
}
}
main()
{
for( ; ; )
{
printf("1.insert_front\t2.delete_front\n");
printf("3.insert_rear\t4.delete_rear\n");
printf("5.display\t6.exit");
printf("enter choice");
scanf("%d",&ch);
                    switch(ch)
{
case 1:printf("enter an item at front end");
    scanf("%d",&item);
    first=insert_front(item,first);
    break;
case 2:first=delete_front(first);
    break;
case 3:printf("enter an item at rear end");
    scanf("%d",&item);
    first=insert_rear(item,first);
    break;
case 4:first=delete_rear(first);
    break;
case 5:display();
    break;
default :exit(0);
}
}
}
```

RESULT APPERED ON SCREEN (I/P & O/P)

```
1.insert_front  2.delete_front
3.insert_rear   4.delete_rear
5.display       6.exit
enter choice

1

enter an item at front end

10

1.insert_front  2.delete_front
3.insert_rear   4.delete_rear
5.display       6.exit
enter choice

2

enter an item at rear end

11

1.insert_front  2.delete_front
3.insert_rear   4.delete_rear
5.display       6.exit
enter choice

2

enter an item at rear end

12

1.insert_front  2.delete_front
3.insert_rear   4.delete_rear
5.display       6.exit
enter choice

5

10→11→12

1.insert_front  2.delete_front
3.insert_rear   4.delete_rear
5.display       6.exit
enter choice

4
```

```
item deleted is 12


1.insert_front  2.delete_front
3.insert_rear   4.delete_rear
5.display       6.exit
enter choice

5

10→11

1.insert_front  2.delete_front
3.insert_rear   4.delete_rear
5.display       6.exit
enter choice

2

item deleted is 10

1.insert_front  2.delete_front
3.insert_rear   4.delete_rear
5.display       6.exit
enter choice

5

11
```

/* Write a program for implementing the operations of double linked list.*/

```
#include<stdio.h>
#include<malloc.h>
int item,ch;
struct node
{
int info;
struct node *rlink;
struct node *llink;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("list empty");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->rlink;
head->rlink=temp;
temp->llink=head;
temp->rlink=cur;
cur->llink=temp;
return head;
}
NODE insert_rear(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
```

```
cur=head->llink;
head->llink=temp;
temp->rlink=head;
temp->llink=cur;
cur->rlink=temp;
return head;
}
NODE delete_front(NODE head)
{
NODE cur,next;
if(head->rlink==head)
{
printf("list empty");
return head;
}
cur=head->rlink;
next=cur->rlink;
head->rlink=next;
next->llink=head;
printf("the node deleted is %d",cur->info);
freenode(cur);
return head;
}

NODE delete_rear(NODE head)
{
NODE cur,prev;
if(head->llink==head)
{
printf("list empty");
return head;
}
cur=head->llink;
prev=cur->llink;
head->llink=prev;
prev->rlink=head;
printf("the node deleted is%d",cur->info);
freenode(cur);
return head;
}
void display(NODE head)
{
NODE temp;
if(head->rlink==head)
{
printf("list is empty");
return;
}
printf("contents of double linked list are\n");
temp=head->rlink;
while(temp!=head)
{
printf("%d\t",temp->info);
temp=temp->rlink;
}
printf("\n");
}
main()
{
NODE head;
head=getnode();
head->rlink=head;
head->llink=head;
for( ; ; )
{
printf("1.insert_front\t2.insert_rear\t3.delete_front\t4.delete_rear\t5.display\t6.exit\n");
printf("enter choice");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("enter the item to be inserted");
       scanf("%d",&item);
       head=insert_front(item,head);
       break;
case 2:printf("enter the item to be insert\n");
       scanf("%d",&item);
       head=insert_rear(item,head);
       break;
case 3:head=delete_front(head);
       break;
case 4:head=delete_rear(head);
       break;
case 5:display(head);
       break;
default:exit(0);
}
}
}
```

RESULT APPERED ON SCREEN (I/P & O/P)

```
Select the choice of operation on link list
1.) insert at begning
2.) insert at at
3.) insert at middle
4.) delete from end
5.) reverse the link list
6.) display list
7.)exit

enter the choice of operation you want to do 1
enter the value you want to insert in node 25
-> 25

enter the choice of operation you want to do 1
enter the value you want to insert in node 56
-> 56 -> 25

enter the choice of operation you want to do 1
enter the value you want to insert in node 66
-> 66 -> 56 -> 25

enter the choice of operation you want to do 2
enter the value you want to insert in node at last 87
-> 66 -> 56 -> 25 -> 87

enter the choice of operation you want to do 2
enter the value you want to insert in node at last 97
-> 66 -> 56 -> 25 -> 87 -> 97

enter the choice of operation you want to do 2
enter the value you want to insert in node at last 99
-> 66 -> 56 -> 25 -> 87 -> 97 -> 99

enter the choice of operation you want to do 3
after which data you want to insert data 25
enter the data you want to insert in list 68
-> 66 -> 56 -> 25 -> 68 -> 87 -> 97 -> 99

enter the choice of operation you want to do 6
-> 66 -> 56 -> 25 -> 68 -> 87 -> 97 -> 99

enter the choice of operation you want to do 7
```

/* Write a program for the representation of polynomials using circular linked list and for the addition of two such polynomials. */

```
#include<stdio.h>
#include<malloc.h>
#include<math.h>
struct node
{
float cf;
float px;
float py;
int flag;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("out of memory\n");
exit(0);
}
return x;
}
NODE insert_rear(float cf,float x,float y,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->cf=cf;
temp->px=x;
temp->py=y;
temp->flag=0;
cur=head->link;
while(cur->link!=head)
cur=cur->link;
cur->link=temp;
temp->link=head;
return head;
}
```

```
void display(NODE head)
{
NODE temp;
if(head->link==head)
{
printf("polinomial does not exist\n");
return;
}
temp=head->link;
while(temp!=head)
{
printf("+%5.2fx^%3.1fy^%3.1f",temp->cf,temp->px,temp->py);
temp=temp->link;
}
printf("\n");
}
NODE add_poly(NODE h1,NODE h2,NODE h3)
{
NODE p1,p2;
int x1,x2,y1,y2,cf1,cf2,cf;
p1=h1->link;
while(p1!=h1)
{
x1=p1->px;
y1=p1->py;
cf1=p1->cf;
p2=h2->link;
while(p2!=h2)
{
x2=p2->px;
y2=p2->py;
cf2=p2->cf;
if(x1==x2&&y1==y2)
break;
p2=p2->link;
}
if(p2!=h2)
{
cf=cf1+cf2;
p2->flag=1;
if(cf!=0)
h3=insert_rear(cf,x1,y1,h3);
}
else
h3=insert_rear(cf1,x1,y1,h3);
p1=p1->link;
}
p2=h2->link;
while(p2!=h2)
{
if(p2->flag==0)
{
h3=insert_rear(p2->cf,p2->px,p2->py,h3);
}
p2=p2->link;
}
return h3;
}
NODE read_poly(NODE head)
{
int i;
float px;
float py;
float cf;
printf("enter the coefficients as -999 to end the polynomial\n");
for(i=1; ;i++)
{
printf("enter the %dterm\n",i);
printf("coeff=");
scanf("%f",&cf);
if(cf==-999)
break;
printf("powx=");
scanf("%f",&px);
printf("powy=");
scanf("%f",&py);
head=insert_rear(cf,px,py,head);
}
return head;
}

main()
{
NODE h1,h2,h3;
h1=getnode();
h2=getnode();
h3=getnode();
h1->link=h1;
h2->link=h2;
h3->link=h3;
printf("enter the first polinomial\n");
h1=read_poly(h1);
```

```
printf("enter the second polinomial\n");
h2=read_poly(h2);
h3=add_poly(h1,h2,h3);
printf("the first polynomial is\n");
display(h1);
printf("the second polynomial is\n");
display(h2);
printf("yhe sum of two polynomial is \n");
display(h3);
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter 1st polynomial
Enter polynomial
Enter -999 to stop
Enter coefficient of 1 term
2
Enter power of x,y:
0
0
Enter coefficient of 2 term
4
Enter power of x,y:
2
2
Enter coefficient of 3 term
6
Enter power of x,y:
3
3
Enter coefficient of 4 term
-999

enter 2nd polynomial
Enter polynomial
Enter -999 to stop

Enter coefficient of 1 term
5
Enter power of x,y:
0
0
Enter coefficient of 2 term
5
Enter power of x,y:
```

```
1
1
Enter coefficient of 3 term
3
Enter power of x,y:
3
3
Enter coefficient of 4 term
-999

1st polynomial

+  2.0x^0y^0+  4.0x^2y^2+  6.0x^3y^3

2nd polynomial

+  5.0x^0y^0+  5.0x^1y^1+  3.0x^3y^3

Addition of two polynomials is

+  7.0x^0y^0+  4.0x^2y^2+  9.0x^3y^3+  5.0x^1y^1
```

/* Write a program for Binary search operations.*/

```
#include<stdio.h>
int bs(int key,int a[],int low,int high)
{
int mid;

if(low>high)
return -1;
mid=(low+high)/2;
if(key==a[mid])
return mid;
if(key>a[mid])
return bs(key,a,mid+1,high);
else
return bs(key,a,low,mid-1);
}
main()
{
int i,n,j,k,ele,temp,a[10];
printf("Enter size of array\n");
scanf("%d",&n);
printf("Enter %d elements\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n;i++)
for(j=0;j<n-1;j++)
{
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
printf("After sorting the elements are   ");
for(i=0;i<n;i++)
printf("%5d",a[i]);
printf("\nEnter the element to be searched   ");
scanf("%d",&ele);
k=bs(ele,a,0,n-1);
if(k==-1)
printf("Element is not found\n");
else
printf("Element is found at the location  %d\n ",k+1);
}
```

RESULT APPERED ON SCREEN (I/P & O/P)

```
Enter size of array
6
Enter 6 elements
1 7 2 3 9 4
After sorting the elements are       1    2    3    4    7    9

Enter the element to be searched   4

Element is found at the location  4

Enter size of array
5
Enter 5 elements
1 7 2 9 3
After sorting the elements are       1    2    3    7    9

Enter the element to be searched   8

Element is not found
```

/*Program to create binary search tree and for implementing inorder,preorder and postorder traversal using recurssion.*/

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *llink;
struct node *rlink;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("out of memory");
exit(0);
}
return x;
}
NODE delete_item(int item,NODE root)
{
NODE cur,parent,suc,q;
if(root==NULL)
{
printf("tree is empty,item not found");
return root;
}
parent=NULL,cur=root;
while(cur!=NULL)
{
if(item==cur->info)break;
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
printf("item not found");
return root;
}
if(cur->llink==NULL)
q=cur->rlink;
else if(cur->rlink==NULL)
```

```
q=cur->llink;
else
{
suc=cur->rlink;
while(suc->llink!=NULL)
suc=suc->llink;
suc->llink=cur->llink;
q=cur->rlink;
}
if(parent==NULL)
return q;
if(cur==parent->llink)
parent->llink=q;
else
parent->rlink=q;
free(cur);
return root;
}
NODE insert_item(int item,NODE root)
{
NODE temp,cur,prev;
temp=getnode();
temp->info=item;
temp->llink=NULL;
temp->rlink=NULL;
if(root==NULL)
return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
if(item==cur->info)
{
printf("duplicate item not allowed\n");
free(temp);
return root;
}
if(item<cur->info)
cur=cur->llink;
else
cur=cur->rlink;
}
if(item<prev->info)
prev->llink=temp;
else
```

```
prev->rlink=temp;
return root;
}
void preorder(NODE root)
{
if(root==NULL)
return;
printf("%d",root->info);
preorder(root->llink);
preorder(root->rlink);
}
void inorder(NODE root)
{
if(root==NULL)
{
return;
}
inorder(root->llink);
printf("%d",root->info);
inorder(root->rlink);
}
void postorder(NODE root)
{
if(root==NULL)
return;
postorder(root->llink);
postorder(root->rlink);
printf("%d",root->info);
}
void search(int item,NODE root)
{
NODE temp;
if(root==NULL)
{
printf("tree is not present\n");
return;
}
temp=root;
while(temp!=NULL)
{
if(item==temp->info)break;
if(item<temp->info)
temp=temp->llink;
else
temp=temp->rlink;
}
```

```
if(temp==NULL)
printf("element is not found\n");
else
printf("element is found\n");
}
main()
{
NODE root=NULL;
int ch,n;
for( ; ; )
{
printf("\n1.insert_item\t2.search\t3.delete_item\t4.inorder\t5.preorder\t6.postorder\t7.exit\n");
printf("enter choice");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("enter element to be inserted");
       scanf("%d",&n);
       root=insert_item(n,root);
       break;
case 2:printf("enter element to be search:");
       scanf("%d",&n);
       search(n,root);
       break;
case 3:printf("enter element to be deleted");
       scanf("%d",&n);
       root=delete_item(n,root);
       break;
case 4:inorder(root);
       break;
case 5:preorder(root);
       break;
case 6:postorder(root);
       break;
default:exit(0);
}
}
}
```

RESULT APPERED ON SCREEN (I/P & O/P)

```
1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
1
Enter element to be insert
12

1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
1
Enter element to be insert
56

1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
1
Enter element to be insert
9

1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
1
Enter element to be insert
78

1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
1
Enter element to be insert
3

1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
4
   3   9  12  56  78
1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
5
  12   9   3  56  78
1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
6
   3   9  78  56  12

1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
```

```
Enter your chioce
2
Enter element to be search
78
Element is found
1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
2
Enter element to be search
99
Element is not found

1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
3
Enter element to be deleted
12
1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
4
   3    9   56   78
1.insert 2.search 3.delete 4.inorder 5.preorder 6.postorder 7.exit
Enter your chioce
7
```

/* Write a program to implement quick sort. */

```
#include<stdio.h>
int partition(int a[],int low,int high)
{
int i,j,key,temp;
key=a[low];
i=low+1;
j=high;
while(1)
{
while(i<high&&key>=a[i])i++;
while(key<a[j])j--;
if(i<j)
temp=a[i],a[i]=a[j],a[j]=temp;
else
{
temp=a[low],a[low]=a[j],a[j]=temp;
return j;
}
}
}
void quicksort(int a[],int low,int high)
{
int j;
if(low<high)
{
j=partition(a,low,high);
quicksort(a,low,j-1);
quicksort(a,j+1,high);
}
}
main()
{
int i,a[20];
int n;
printf("enter the value of n");
scanf("%d",&n);
printf("enter the numbers to be sorted");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
quicksort(a,0,n-1);
printf("the sorted array is\n");
for(i=0;i<n;i++)
printf("%d",a[i]);
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter size of an array
5
Enter array elements
1 7 2 8 3

The elements after sorting are      1    2    3    7    8
```

/* Write a program to implement Heap sort.*/

```
#include<stdio.h>
void heapify(int a[],int n)
{
int i,j,k,item;
for(k=1;k<n;k++)
{
item=a[k];
i=k;
j=(i-1)/2;
while(i>0&&item>a[j])
{
a[i]=a[j];
i=j;
j=(i-1)/2;
}
a[i]=item;
}
}
void adjust(int a[],int n)
{
int i,j,item;
j=0;
item=a[j];
i=2*j+1;
while(i<=n-1)
{
if(i+1<=n-1)
if(a[i]<a[i+1])i++;
if(item<a[i])
{
a[j]=a[i];
j=1;
i=2*j+1;
}
else
break;
}
a[j]=item;
}
void heapsort(int a[],int n)
{
int i,temp;
heapify(a,n);
for(i=n-1;i>0;i--)
{
temp=a[0],a[0]=a[i],a[i]=temp;
adjust(a,i);
}
}
main()
{
int a[20],n,temp,i;
printf("enter the no.of elements to sort\n");
scanf("%d",&n);
printf("enter the elements to sort\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
heapsort(a,n);
printf("the sorted arrayis \n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter size of array

5

Enter the elements in to the array

1 7 2 3 9


heap array:
9       7       2       1       3
 The sorted array is:1  2       3       7       9
 Complexity: best case=avg case=worst case=o(n logn)
```

/* Write a program to implement Merge sort.*/

```
#include<stdio.h>
#define MAX 1000
void simple_merge(int a[],int low,int mid,int high)
{
int i=low;
int j=mid+1;
int k=low;
int c[MAX];
while(i<=mid&&j<=high)
{
if(a[i]<a[j])
{
c[k]=a[i];
i=i+1;
k=k+1;
}
else
{
c[k]=a[j];
j=j+1;
k=k+1;
}
}
while(i<=mid)
{
c[k++]=a[i++];
}
while(j<=high)
{
c[k++]=a[j++];
}
for(i=low;i<=high;i++)
{
a[i]=c[i];
}
}
void merge_sort(int a[],int low,int high)
{
int mid;
if(low<high)
{
mid=(low+high)/2;
merge_sort(a,low,mid);
merge_sort(a,mid+1,high);
simple_merge(a,low,mid,high);
}
}
main()
{
int n,i;
int a[MAX];
printf("enter the number of elements to sort\n");
scanf("%d",&n);
printf("enter the elementsto sort\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
merge_sort(a,0,n-1);
printf("the sorted vector is\n");
for(i=0;i<n;i++)
printf("%d",a[i]);
}
```

RESULT APPERED ON SCREEN (I/P & O/P)

```
Enter the size of an array

5

Enter elements in to an array

1

7

2

8

3


After sorting the elements are    1   2   3   7   8
```

./***Simple insertion sort implementation using array s in ascending order***/

```
#include<stdio.h>

int main(){

  int i,j,s,temp,a[20];

  printf("Enter total elements: ");
  scanf("%d",&s);

  printf("Enter %d elements: ",s);
  for(i=0;i<s;i++)
      scanf("%d",&a[i]);

  for(i=1;i<s;i++){
      temp=a[i];
      j=i-1;
      while((temp<a[j])&&(j>=0)){
      a[j+1]=a[j];
          j=j-1;
      }
      a[j+1]=temp;
  }

  printf("After sorting: ");
  for(i=0;i<s;i++)
      printf(" %d",a[i]);

  return 0;
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter total elements: 5

Enter 5 elements: 3 7 9 0 2

After sorting:  0 2 3 7 9
```

/***Simple Selection sort implementation using array ascending** */

```
#include<stdio.h>
int main(){

  int s,i,j,temp,a[20];

  printf("Enter total elements: ");
  scanf("%d",&s);

  printf("Enter %d elements: ",s);
  for(i=0;i<s;i++)
      scanf("%d",&a[i]);

  for(i=0;i<s;i++){
      for(j=i+1;j<s;j++){
           if(a[i]>a[j]){
               temp=a[i];
              a[i]=a[j];
              a[j]=temp;
           }
      }
  }

  printf("After sorting is: ");
  for(i=0;i<s;i++)
      printf(" %d",a[i]);

  return 0;
}
```

RESULT APPERED ON SCREEN (I/P & O/P)

```
Enter total elements: 5

Enter 5 elements: 4 5 0 21 7

The array after sorting is:  0 4 5 7 21
```

/*Program for sorting array using shell sorting method*/

```
#include<stdio.h>
#include<conio.h>
int main()
{
 int arr[30];
 int i,j,k,tmp,num;
 printf("Enter total no. of elements : ");
 scanf("%d", &num);
 for(k=0; k<num; k++)
 {
   printf("\nEnter %d number : ",k+1);
   scanf("%d",&arr[k]);
 }
 for(i=num/2; i>0; i=i/2)
 {
   for(j=i; j<num; j++)
   {
     for(k=j-i; k>=0; k=k-i)
     {
        if(arr[k+i]>=arr[k])
            break;
        else
        {
            tmp=arr[k];
            arr[k]=arr[k+i];
            arr[k+i]=tmp;
        }
     }
   }
 }
 printf("\t**** Shell Sorting ****\n");
 for(k=0; k<num; k++)
     printf("%d\t",arr[k]);
 getch();
 return 0;
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter total no. of elements : 7
Enter 1 number : 8
Enter 2 number : 3
Enter 3 number : 7
Enter 4 number : 9
Enter 5 number : 1
Enter 6 number : 24
Enter 7 number : 2

 **** Shell Sorting ****
1  2  3  7  8  9  24
```

/***Program to Traverse the Tree Non-Recursively** */

```
#include <stdio.h>
#include <stdlib.h>

  struct node
  {
      int a;
      struct node *left;
      struct node *right;
};

  void generate(struct node **, int);
  int search(struct node *, int);
  void delete(struct node **);

  int main()
  {
      struct node *head = NULL;
      int choice = 0, num, flag = 0, key;

     do
      {
        printf("\nEnter your choice:\n1. Insert\n2. Search\n3. Exit\nChoice: ");
         scanf("%d", &choice);
         switch(choice)
         {
         case 1:
             printf("Enter element to insert: ");
            scanf("%d", &num);
             generate(&head, num);
             break;
         case 2:
         printf("Enter key to search: ");
            scanf("%d", &key);
           flag = search(head, key);
             if (flag)
        {
                printf("Key found in tree\n");
             }
            else
            {
                printf("Key not found\n");
           }           break;          case 3:
            delete(&head);
            printf("Memory Cleared\nPROGRAM TERMINATED\n");
            break;
        default: printf("Not a valid input, try again\n");
         }
     } while (choice != 3);
   return 0;
}

void generate(struct node **head, int num)
{
    struct node *temp = *head, *prev = *head;

    if (*head == NULL)
    {
        *head = (struct node *)malloc(sizeof(struct node));
        (*head)->a = num;
        (*head)->left = (*head)->right = NULL;
    }
    else
   {
        while (temp != NULL)
        {
            if (num > temp->a)
           {
                prev = temp;
                temp = temp->right;
           }
            else
            {
                prev = temp;
                temp = temp->left;
            }
        }
        temp = (struct node *)malloc(sizeof(struct node));
       temp->a = num;
        if (num >= prev->a)
        {
            prev->right = temp;
        }
        else
        {
            prev->left = temp;
        }
    }
}
```

```
 int search(struct node *head, int key)
 {
     while (head != NULL)
     {
         if (key > head->a)
         {
             head = head->right;
       }
         else if (key < head->a)
         {
             head = head->left;
         }
      else
         {
           return 1;
         }
    }
             return 0;
}

void delete(struct node **head)
{
    if (*head != NULL)
   {
        if ((*head)->left)
        {
            delete(&(*head)->left);
         }
         if ((*head)->right)
         {
            delete(&(*head)->right);
        }
      free(*head);
    }
 }
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter your choice:
1. Insert
2. Search
3. Exit
Choice: 1
Enter element to insert: 1

Enter your choice:
1. Insert
2. Search
3. Exit
Choice: 1
Enter element to insert: 2

Enter your choice:
1. Insert
2. Search
3. Exit
Choice: 1
Enter element to insert: 3

Enter your choice:
1. Insert
2. Search
3. Exit
Choice: 1
Enter element to insert:
4

Enter your choice:
1. Insert
2. Search
3. Exit
Choice: 2
Enter key to search: 1
Key found in tree

Enter your choice:
1. Insert
2. Search
3. Exit
Choice: 2
Enter key to search: 6
Key not found
```

/***Program to Implement Priority Queue to Add and Delete Elements***/

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
void insert_by_priority(int);
void delete_by_priority(int);
void create();
void check(int);
void display_pqueue();

 int pri_que[MAX];
  int front, rear;

  void main()
  {
      int n, ch;

      printf("\n1 - Insert an element into queue");
      printf("\n2 - Delete an element from queue");
      printf("\n3 - Display queue elements");
 printf("\n4 - Exit");
   create();
    while (1)

{

printf("\nEnter your choice : ");

  scanf("%d", &ch);
          switch (ch)
         {
          case 1:
             printf("\nEnter value to be inserted : ");
             scanf("%d",&n);
              insert_by_priority(n);
             break;
          case 2:
             printf("\nEnter value to delete : ");
              scanf("%d",&n);
              delete_by_priority(n);
              break;
           case 3:
              display_pqueue();
              break;
```

```
        case 4:
            exit(0);
       default:
            printf("\nChoice is incorrect, Enter a correct choice");
        }
    }
    }

/* Function to create an empty priority queue */
  void create()
   {
   front = rear = -1;
}

/* Function to insert value into priority queue */
void insert_by_priority(int data)
 {
   if (rear >= MAX - 1)
   {
     printf("\nQueue overflow no more elements can be inserted");
       return;
       }
    if ((front == -1) && (rear == -1))
    {
        front++;
        rear++;
        pri_que[rear] = data;
        return;
    }
    else
        check(data);
    rear++;
}

/* Function to check priority and place element */
void check(int data)
{
    int i,j;

    for (i = 0; i <= rear; i++)
    {
        if (data >= pri_que[i])
        {
            for (j = rear + 1; j > i; j--)
            {
                pri_que[j] = pri_que[j - 1];
            }
            pri_que[i] = data;
            return;
        }
    }
    pri_que[i] = data;
}

/* Function to delete an element from queue */
void delete_by_priority(int data)
{
    int i;

    if ((front==-1) && (rear==-1))
    {
        printf("\nQueue is empty no elements to delete");
        return;
    }

    for (i = 0; i <= rear; i++)
    {
        if (data == pri_que[i])
        {
            for (; i < rear; i++)
            {
              pri_que[i] = pri_que[i + 1];
    }

        pri_que[i] = -99;
        rear--;

        if (rear == -1)
            front = -1;
        return;
        }
    }
    printf("\n%d not found in queue to delete", data);
}
/* Function to display queue elements */
void display_pqueue()
  {
   if ((front == -1) && (rear == -1))
    {
     printf("\nQueue is empty");
       return;
      }
    for (; front <= rear; front++)
    {
          printf(" %d ", pri_que[front]);

    }

    front = 0;
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
1 - Insert an element into queue
2 - Delete an element from queue
3 - Display queue elements
4 - Exit
Enter your choice : 1

Enter value to be inserted : 20

Enter your choice : 1

Enter value to be inserted : 45

Enter your choice : 1

Enter value to be inserted : 89

Enter your choice : 3
 89  45  20
Enter your choice : 1

Enter value to be inserted : 56

Enter your choice : 3
 89  56  45  20
Enter your choice : 2

Enter value to delete : 45

Enter your choice : 3
 89  56  20
Enter your choice : 4
```

/* program for finding the bfs and dfs of a graph.*/

```
 #include<stdio.h>
int q[20],top=-1,rear=-1,front=-1,a[20][20],vis[20],stack[20];
int delete();
void add(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();
main()
{
int i,n,s,ch,j;
char c,dummy;
printf("enter no.of vertices\n");
scanf("%d",&n);
printf("enter adjacency matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
for( ; ; )
{
printf("\n menu");
printf("\n 1.bfs");
printf("\n 2.dfs");
printf("\nexit");
printf("\nenter your choice");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("enter the source vertex\n");
        scanf("%d",&s);
        bfs(s,n);
        break;
case 2:printf("enter source vertex\n");
        scanf("%d",&s);
        dfs(s,n);
        break;
case 3:exit(0);
        break;
}
}
}
void bfs(int s,int n)
{
int p,i;
for(i=1;i<=n;i++)
```

```
vis[i]=0;
add(s);
vis[s]=1;
p=delete();
if(p!=0)
printf("%d\t",p);
while(p!=0)
{
for(i=1;i<=n;i++)
if((a[p][i]!=0)&&(vis[i]==0))
{
add(i);
vis[i]=1;
}
p=delete();
if(p!=0)
printf("%d\t",p);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
bfs(i,n);
}
void add(int item)
{
if(rear==19)

printf("queue is full\n");
else
{
if(rear==-1)
{
q[++rear]=item;
front++;
}
else
q[++rear]=item;
}
}
int delete()
{
int k;
if((front>rear)||(front==-1))
return(0);
else
{
k=q[front++];
```

```
return(k);
}
}
void dfs(int s,int n)
{
int i,k;
for(i=1;i<=n;i++)
vis[i]=0;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf("%d\t",k);
while(k!=0)
{
for(i=1;i<=n;i++)
if((a[k][i]!=0)&&(vis[i]==0))
{
push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf("%d\t",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);

}
void push(int item)
{
if(top==19)
printf("stack overflow");
else
stack[++top]=item;
}
int pop()
{
int k;
if(top==-1)
return 0;
else
{
k=stack[top--];
return(k);
```

```
}
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter number of vertices
5
Enter adjacency matrix
0 1 1 1 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 1
0 0 0 0 0
Menu
1.BFS
2.DFS
3.EXIT
Enter your choice    1
Enter the source vertex 1
The BFS Traversal is
1 2 3 4 5
Menu
1.BFS
2.DFS
3.EXIT
Enter your choice    2
Enter the source vertex 1
The DFS Traversal is
1 4 5 3 2
Menu
1.BFS
2.DFS
3.EXIT
Enter your choice    3
```

/* program to find transistive closure of a given graph.*/

```
include<stdio.h>
#include<math.h>
void warshall(int p[10][10],int n)
{
int i,j,k;
for(k=1;k<=n;k++)
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
p[i][j]=p[i][j]||(p[i][k]&&p[k][j]);
}
main()
{
int p[10][10]={0},n,i,j;
printf("\n enter number of vertices");
scanf("%d",&n);
printf("enter adjacency matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&p[i][j]);
warshall(p,n);
printf("\n transistive closure for the given graph is \n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
printf("%d\t",p[i][j]);
printf("\n");
}
}
```

<u>RESULT APPERED ON SCREEN (I/P & O/P)</u>

```
Enter number of vertices 4

Enter adjacency matrix

0 1 0 0

0 0 1 0

1 0 0 1

0 0 0 0

Transitive closure for the given graph is

1       1       1       1

1       1       1       1

1       1       1       1

0       0       0       0
```