SECTION – 1

Data and File Structures

Session 1 : Arrays

Ex 1: Write a program in `C' language that accepts two matrices as input and prints their product.

Code:

```
#include <stdio.h>
#define MAX 10
void main()
{
        int a[MAX][MAX],b[MAX][MAX],c[MAX][MAX],i,j,k,m,n,p,q;
        clrscr();
        printf("Enter the order of the first matrix\n");
        scanf("%d %d",&m,&n);
        printf("Enter the order of the second matrix\n");
        scanf("%d %d",&p,&q);
        if(n!=p)
        {
        printf("The  matrix can not be multiplied\n");
        }
        else
        {
        printf("Enter the elements of the first matrix\n");
        for(i=0;i<m;i++)
        for(j=0;j<n;j++)
        scanf("%d",&a[i][j]);
        printf("Enter the elements of the second matrix\n");
        for(i=0;i<p;i++)
        for(j=0;j<q;j++)
        scanf("%d",&b[i][j]);
        for(i=0;i<m;i++)
        for(j=0;j<q;j++)
        {
        c[i][j]=0;
        for(k=0;k<n;k++)
        c[i][j]=c[i][j]+(a[i][k]*b[k][j]);
        }
        printf("The  resultant matrix on multiplication is\n");
        for(i=0;i<m;i++)
        {
        for(j=0;j<q;j++)
        printf("%d\t",c[i][j]);
        printf("\n");
        }
        }
        getch();
 }
```

Output:

```
Enter the order of the first matrix
2 3
Enter the order of the second matrix
3 2
Enter the elements of the first matrix
1 2
3 4
5 6
Enter the elements of the second matrix
1 2 3
4 5 6
The resultant matrix on multiplication is
22 28
49 64
```

Ex 2: Write a program in `C' Language to accept 10 strings as input and print them in lexicographic order.

Code:

```
#include <stdio.h>
void main()
{
        char str[10][10],t[10];
        int i,j;
        clrscr();
        for(i=0;i<10;++i)
        strcpy(str[i],"");
        printf("Enter the strings -\n");
        for(i=0;i<10;++i)
        scanf("%s",str[i]);
        for(i=0;i<10;++i)
        {
        for(j=i+1;j<10;++j)
        if(strcmp(str[i],str[j])>0)
        {
        strcpy(t,str[i]);
        strcpy(str[i],str[j]);
        strcpy(str[j],t);
        }}
        printf("The strings in lexicographical order is -\n");
        for(i=0;i<10;++i)
        printf("%s\n",str[i]);
        getch();
 }
```

Output:

```
Enter 10 strings -
Irshad Amol Abhay Pranay Anant Mangesh Vishal Alok Vivek Sameer

Strings in lexicographic order-
Abhay Amol Anant Alok Irshad Mangesh Pranay Sameer Vishal Vivek
```

Ex 3: Write a program in `C' Language that accepts two strings S1 and S2 as input.
The program should check if S2 is a substring of S1 or not. If S2 is a substring of S1, then the program should output the starting location and ending location of S2 in S1. If S2 appears more than once in S1, then the locations of all instances have to be given.

Code:

```
#include
#include<string.h>
void main()
  {
        int i=0,j=0,v,c=0,l1,l2;
        char s1[50],s2[50];
        clrscr();
        printf("Enter 2 strings\n");
        gets(s1);
        gets(s2);
        l1=strlen(s1);
        l2=strlen(s2);
        while(i<=l1)
        {
        if(j==l2)
        {
        v=i-l2+1;
        c++;
        printf("Start of %d occurance=%d end =%d\n",c,v,i);
        j=0;
        }
        else if(s1[i]==s2[j])
        {
        i++;
```

```
                j++;
                }
                else
                {
                i++;
                j=0;
                }
                }
                if(c==0)
                printf("Not substring");
                getch();
        }
```

Output:

```
Enter 2 strings -
                welcome to alfec welcome to all
                welcome
Start of 1 occurance=1 end =7 Start of 2 occurance=18 end =24
```

Ex 4: Write a program to concatenate two strings S1 and S2.

Code:

```
#include<stdio.h>
#include<conio.h>
void main()
   {
                char *s1, *s2, *s3;
                int length, len1=0,len2=0, i, j;
                clrscr();
                s1=(char*)malloc(20* sizeof(s1));
                s2=(char*)malloc(20* sizeof(s2));
                printf("Enter first string\n");
                gets(s1);
                len1=strlen(s1);
                printf("Enter second string\n");
                gets(s2);
                len2=strlen(s2);
                length=len1+len2;
                s3= (char*)malloc((length+2)* sizeof(s3));
                for(i=0;i<len1;++i)
                *(s3+i) =*(s1+i);
                *(s3+i)=' '; /*leave a space at end of first string */
                ++i;
                for(j=0;j<len2;++j)
                { *(s3+i)=*(s2+j); /* copying 2nd string */
                   ++i;
                }
                *(s3+i)='\0'; /* store '\0' at end to set 'end of string' */
                printf("Concatenated string is\n%s", s3);
                getch();
        }
```

Output:

```
Enter first string : Welcome
Enter second String: to Alfec
concatenated string : Welcome to Alfec
```

Session 2 : Structures

Ex 5: Write a program in 'C' language, which accepts Enrolment number, Name Aggregate marks secured in a Program by a student. Assign ranks to students according to the marks secured. Rank-1 should be awarded to the students who secured the highest marks and so on. The program should print the enrolment number, name of the student and the rank secured in ascending order.

Code:

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

```
struct stud
   {
int roll;
int mark;
char name[15];
 };
main()
   {
        struct stud S[10], t;
        int i, j, n;
        clrscr();
        printf("Enter numbers of students\n");
        scanf("%d",&n);
        for(i=0;i<n;++i)
        {
        printf("Enter roll no., name, mark\n");
        scanf("%d %s %d",&S[i].roll,&S[i].name,&S[i].mark);
        }
        for(i=0;i<n;i++)
        for(j=i+1;j<3;++j)
        if(S[i].mark<S[j].mark)
        {
        if(S[i].mark<S[j].mark)
        {
        t=S[i];
        S[i]=S[j];
        S[j]=t;
        }
        printf("\nRank List\n\n");
        printf("Roll No Name Mark Rank\n");
        for(i=0;i<n;++i)
        {
        printf("%d\t%s\t\t%d\t%d\n",S[i].roll,S[i].name,S[i].mark,i+1);
        }
        }
        getch();
   }
```

Output:

```
Enter numbers of students
3
Enter roll no., name, mark
1
VINU
50
Enter roll no., name, mark
2
GEETHA
60
Enter roll no., name, mark
3
RAMU
55
Rank List
Roll No      Name      Mark     Rank
   2       GEETHA      60        1
   1         VINU      50        2
   3         RAMU      55        3
```

Ex 6: Write a program in 'C' language to multiply two sparse matrices.

Code:

```
        /*to multiply two sparce matrices***/
#include<stdio.h>
#include<conio.h>
        /*function to convert to a sparse matrix ***/
```

```
struct sp
   {
         int row,col;
         int mat[10][10];
         int sp[50][3];
   };
         /* to convert the entered matrix to sparce form,by eleminating zero values*/
int convsp(struct sp *M )
{
int e=1,i,j;
printf("enter number of rows and columns in the matrix");
scanf("%d%d",&M->row,&M->col);
printf("enter the matrix");
for(i=0;i<M->row;++i)
for(j=0;j<M->col;++j)
{scanf("%d",&M->mat[i][j]);
if(M->mat[i][j]!=0)
{
M->sp[e][0]=i;
M->sp[e][1]=j;
M->sp[e][2]=M->mat[i][j];
e++;
}//end if
}/*end j loop*/
M->sp[0][0]=M->row; //store number of rows to first(row,col) of sparse
M->sp[0][1]=M->col; //store number of cols to first row,second col
M->sp[0][2]=e-1; //store total number of non zero values to 3rd column
return M->sp[0][2]; //return total number of non zero elements
}        /*to multiply the 2 matrix**/
mult(struct sp M1,int e1, struct sp M2,int e2)
{
int sum[10][10],i,j,k1,k2;
for(i=0;i<10;++i)
for(j=0;j<10;++j)
sum[i][j]=0;
for(i=1;i<=e1;++i)
for(j=1;j<=e2;++j)
if(M1.sp[i][1]==M2.sp[j][0])
{
k1=M1.sp[i][0]; k2=M2.sp[j][1];
sum[k1][k2]+=M1.sp[i][2]*M2.sp[j][2];
}
printf("\nproduct matrix\n");
for(i=0;i<M1.row;++i)
{
for(j=0;j<M2.col;++j)
printf("%d\t",sum[i][j]);
printf("\n");
}
}        /*to print sparse matrix ***/
void printsp(int n,struct sp matx)
{
int i,j;
for(i=0;i<=n;++i)
{
for(j=0;j<3;++j)
printf("%d\t",matx.sp[i][j]);
printf("\n");
}}
main()
   {
         int ele1,ele2;
         struct sp m1,m2;
         clrscr();
         ele1=convsp(&m1);
```

```
        printf("\n SPARSE MATRIX1\n");
        printsp(ele1,m1);
        ele2=convsp(&m2);
        printf("\n SPARSE MATRIX2\n");
        printsp(ele2,m2);
        if(m1.row!=m2.col)
        printf("matrices are not compatible for multiplication");
        else
        mult(m1,ele1,m2,ele2);
        getch();
    }
```

Output:

```
Enter number of rows and columns in the matrix
3 3
Enter the matrix
1 0 0
0 1 0
1 1 1
SPARSE MATRIX1
3 3 5
0 0 1
1 1 1
2 0 1
2 1 1
2 2 1
Enter number of rows and columns in the matrix
3 3
Enter the matrix
0 1 0
1 0 0
1 0 1
SPARSE MATRIX2
3 3 4
0 1 1
1 0 1
2 0 1
2 2 1
Product matrix
0 1 0
1 0 0
2 1 1
```

Ex 7: Write a program in 'C' language to accept a paragraph of text as input. Make a list of words and the number of occurrences of each word in the paragraph as output. As part of the processing, an array and structure should be created wherein each structure consists of two fields, namely, one for storing the word and the other for storing the number of occurrences of that word.

Code:

```
#include<stdio.h>
#include<conio.h>
struct paragraph
{
char words[15];
int occ;
}p[50];
void main()
{
int i=0,j=0,k=0,flag=0;
char w[15],ch=0;
clrscr();
strcpy(w," ");
printf("Enter the paragraph\n");
while(ch!='\n')
{
flag=0;
```

```
strcpy(p[j].words,"");
p[j].occ=1;
ch=getchar();
w[i]=ch;
i++;
if(ch==' ')
{
w[i]='\0';
for(k=0;k<=j;++k)
if(strcmp(p[k].words,w)==0)
{
p[k].occ++;
flag=1;
}
if(flag==0)
{
strcpy(p[j].words,w);
++j;
}
strcpy(w," ");
i=0;
}
}
printf("words\t\toccurance\n");
for(i=0;i<j;i++)
printf("%s\t\t%d\n",p[i].words,p[i].occ);
getch();
}
```

Output:

```
Enter a paragraph
May God bless you children be good children
Word Occurrence
May 1
God 1
bless 1
you 1
children 2
good 1
be 1
```

Session 3 : Linked Lists

Ex 8: Write a program in 'C' language for the creation of a list. Also, write a procedure for deletion of an element from the list. Use pointers.

Code:

```
/*Creation and deletion of linked list*/
#define NULL 0
struct student
{
char name[15];
int roll_no;
struct student *next;
}*stud,*first;
/*creation of list*/
list_create(struct student *s1)
{
printf("Enter roll number:-1 to terminate\n");
scanf("%d",&s1->roll_no);
if(s1->roll_no!=-1)
{
printf("Enter name: ");
scanf("%s",s1->name);
s1->next=(struct student*)malloc(sizeof(struct student));
list_create(s1->next);
}
```

```
else
{
s1->next=NULL;
return;
}
}
/*Display the list */
display_list(struct student *s1)
{
if(first->next==NULL)
{
printf("List is empty");
getch();
return 0;
}
while(s1->next)
{
printf("%d\t%s\n",s1->roll_no,s1->name);
s1=s1->next;
}
getch();
return 0;
}
/*Delete from list */
delete_element(struct student *start)
{
struct student *temp,*t;
int roll,flag=0;
if(first->next==NULL)
{
printf("List empty");
getch();
return 0;
}
printf("Enter rollnumber to delete\n");
scanf("%d",&roll);
if(start->roll_no==roll)
{
temp=start->next;
free(start);
first=temp;
return 0;
}
/* any other node */
t=start;/*store previous node t*/
start=start->next; /*point next node */
while(start)
{
if(start->roll_no==roll)
{
temp=start->next;
free(start);
t->next=temp;
return;
}
t=t->next;
start=start->next;
flag=1;
}
if(flag==1)
{
printf("Rollnumber not found");
getch();
return(0);
}
```

```
}
main()
{
int choice=0;
clrscr();
stud=(struct student*)malloc(sizeof(struct  student));
first=stud;
first->next=NULL;
while(choice!=4)
{
clrscr();
printf("MENU\n\n");
printf("Create....1\n\nDisplay...2\n\nDelete...3\n\nExit...4\n\n");
printf("Enter choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:
list_create(stud);
break;
case 2:
display_list(stud);
break;
case 3:
delete_element(stud);
break;
case 4: break;
}
}
getch();
return 0;
}
```

Output:

```
MENU
        67. create
        68. display
        69. delete
        70. Exit
Enter choice : 1
Enter roll number : 12
Enter name : Anu
Enter roll number : -1
return to main menu
```

Ex 9: Write a program in 'C' language that accepts two singly linked lists A and B as input. Now, print a singly linked list that consists of only those elements, which are common to both A and B.

Code:

```
# include<stdio.h>
# include<conio.h>
struct student
{
int roll no; struct student *next;} *s1, *s2;
/*creation of linked list*/
List-create (struct student *S1)
{
print f ("enter roll number") ; Scan f("%d", & s1®roll no.);
if (s1®roll no 1= -1) /* '-1' is entered to stop*/
{
si®next (struct student*) malloc (size of (struct students));
list-create (s1®next);
}
else s1®next = NULL
return
}
```

```
/*display the list*/
void display list (struct sudent *s1)s
{
if (s®next l== NULL)
{
print f (“ %d \n”, s1®roll no);
display – list (s1®next)
}
}
/* function to print common elements */
common-elements (styruct student *s1, struct student * s2)
(
int flag = 0
struct student * temp s2; /* temp is to store the initial node of s2*)
print f (common elements \n);
while (s1l®next 1=NULL) /*repeat till the end of s1*/
{
while (s2®next 1=NULL)/* This loop is repeated that many times the number of nodes in s1x s2
*/
{
if (s1®roll no==s2®roll no.)
{
flag=1; print f (“%d \n”, s1®roll no.);
/*flag is set to show that an equal roll number is met*/
}
s2=s2®next;
}
s1=s1®next; s2=templ /* s2 is to be started again from beginning when s1=s1®next */
}
if (flag=0)
print f (“no common elements”)`
}
main ( )
{
s1=(struct student*) malloc (struct student));
s2=(struct student*) mulloc (size of (structstudent));
/*create the ‘s1’ list */
list-create (s1);
print f (“elements in first list \n”); display- list (s1)
get ch ( )
print f (“enter elements to second list”);
list-create (s2); /* creating second list */
print f(“elements in second list \n”) display-list (s2);
/*printing common elements */
common-element (s1, s2) get ch( );
}
```

Output:

```
enter roll number : 12
enter roll number : 13
enter roll number : 14
enter roll number : -1
elements in the first list :
12 13 14

enter elements to second list
enter roll number : 25
enter roll number : 14
enter roll number : 26
enter roll number : 13
enter roll number : -1

Elements in the second list
25 14 26 13
Common elements
```

Ex 10: Write a program in 'C' language to accept a singly linked list of integers as input. Now, sort the elements of the list in ascending order. Then, accept an integer as input. Insert this integer into the singly linked list at the appropriate position.

Code:

```
/* to sort a linked list and insert an element at the proper position*/
#include<stdio.h>
#include<conio.h>
struct student
{
int rollno;
struct student *next;
}*stud,*first;
/*******creation of list********/
void list_create(struct student *s1)
{
clrscr();
printf("enter roll number-1 to stop"); scanf("%d",&s1->rollno);
if(s1->rollno!=-1)
{
// printf("enter name"); scanf("%s",s1->name);
s1->next=(struct student*)malloc(sizeof(struct student));
list_create(s1->next);
}
else s1->next=NULL;
return;
}
/*****display the list **********/
void display_list(struct student *s1)
{
if(s1->next!=NULL)
{
printf("%d\n",s1->rollno);
// printf("%d\t%s",s1->next->rollno,s1->next->name);
// printf("%d\t%s",s1->next->nextrollno,s1->next->next->name);
display_list(s1->next);
}
}
/********sort list ***********/
void sort_list(struct student *s1)
{
struct student *temp,*t,*t1;
if(s1->next==NULL)
return;
t=s1;
while(t)
{
t1=t->next;
while(t1->next!=NULL)
{if(t->rollno>t1->rollno)
{temp->rollno=t->rollno;
t->rollno=t1->rollno;
t1->rollno=temp->rollno;
}t1=t1->next;}
t=t->next;}
}
/**inserting an element to list*/
insert_element(struct student *s1)
{ int r; struct student* temp,*prev;
printf("enter rollnumber to insert"); scanf("%d",&r);
//to insert before the first node
if(r<s1->rollno)
{ temp=(struct student*)malloc(sizeof(struct student));
temp->rollno=r;
temp->next=s1;
first=temp; return;
```

```
}
/*to insert in between any node*/
while(s1->next)
{
if(s1->rollno <r){prev=s1;
s1=s1->next;}
else
{temp=(struct student*)malloc(sizeof(struct student));
temp->rollno=r;
temp->next=prev->next;
prev->next=temp;
break;}
}
/*to insert after last node*/
if(s1->next==NULL && r>s1->rollno)
{
temp=(struct student*)malloc(sizeof(struct student));
temp->rollno=r;
temp->next=prev->next;
prev->next=temp;
}}
/********searching for an element in the list ********/
/*struct student* search(struct student *start, int rn)
{
if(start->next==NULL)
return (NULL);
if(start->next->rollno==rn)
return(start);
else
search(start->next,rn);
return NULL;
}*/
/********* delete element from list *************/
/*struct student* delete_element(struct student *start)
{
struct student *temp,*t; int roll;
printf("enter rollnumber to delete"); scanf("%d",&roll);
if(start->rollno==roll)
{
temp=start->next;
free(start);
start=temp;
}
else
{
t=search(start,roll);
if(t==NULL)
printf("roll number not found\n");
else
{ temp=t->next->next; free(t->next); t->next=temp; }
}return(start);
}*/
/*********main ***********/
main()
{
clrscr();
first=(struct student*)malloc(sizeof(struct student));
stud=(struct student*)malloc(sizeof(struct student));
first=stud;// first->next=NULL;
list_create(stud);
display_list(stud);
//stud=delete_element(stud);
printf("\nsorted list\n");
sort_list(stud);
display_list(stud);
```

```
insert_element(stud);
display_list(first);
getch();
return;
}
```

Output:

```
Enter roll number-1  to stop
10 68 74 1 22 99 4 3

Sorted list -
1  3  4  10  22  68  74  99

Enter rollnumber to insert
15 1 3 4 10
```

Session 4 : Stacks

Ex 12: Write a program in 'C' language to reverse an input string.

Code:

```
/* to reverse  a string using stack */
#include<stdio.h>
#include<conio.h>
struct list
{
char ch;
struct list *next;
}*top=NULL;
/*store string to stak*/
push(char s)
{
struct list* t;
t=(struct list*)malloc(sizeof(struct list));
t->ch=s;
t->next=top;
top=t;
}
/*display reverse string*/
display()
{
struct list *tmp;
tmp=top;
while(tmp!=NULL){
printf("%c",tmp->ch);
tmp=tmp->next;
}
}
main()
{
char c;
clrscr();
printf("enter a string\n");
while(c!='\n')
{
c=getchar();
push(c); }
printf("reversed string is\n");
display();
getch();
}
```

Output:

```
Enter a string : IGNOU
Reversed string is - UONGI
```

Ex 13: Write a program in 'C' language to implement multiple stacks in a single array.

Code:

```
/*multiple stack in a single array */
#include<stdio.h>
#include<conio.h>
int sno,t1,t2;
int a[20];
void push(int,int);
void pop(int);
void main()
{
int val,t=1,no;
clrscr();
t1=0;
t2=10;
while(t)
{
printf("1.push\n2.pop\n3.exit\n");
printf("enter your choice");
scanf("%d",&t);
switch(t)
{
case 1:
printf("enter the stack no:\n");
scanf("%d",&sno);
printf("enter the value:\n");
scanf("%d",&val);
push(sno,val);
break;
case 2:
printf("enter the stack no:\n");
scanf("%d",&sno);
pop(sno);
break;
case 3:
exit();
}
}
getch();
}
void push(int sno,int val)
{
switch(sno)
{
case 1:
if((++t1)==10)
printf("stack 1:overflow\n");
else
a[t1]=val;
break;
case 2:
if(++t2==20)
printf("stack 2:overflow\n");
else
a[t2]=val;
break;
}
}
void pop(int sno)
{
switch(sno)
{
case 1:
if(t1<=0)
printf("stack 1:empty\n");
```

```
else
{
printf("%d\n",a[t1]);
t1--;
}
break;
case 2:
if(t2<=10)
printf("stack 2:empty\n");
else
{
printf("%d\n",a[t2]);
t2--;
}
break;
}
}
```

Output:

```
1. push
2. pop
3. exit
enter your choice :1
enter stuck no.1
enter value  10
1. push
2. pop
3. exit
enter your choice :1 enter stuck no.1
enter value:  11
enter choice : 1 enter stuck no.2
enter value  : 21
enter choice : 1, enter stuck no.2
enter value  : 22
enter choice :2 enter stuck no.1
value  = 11
enter choice : 2 ENTER STUCK NO.2
value  = 22
enter choice : 3
exit the program
```

Session 5 : Queues

Ex 15: Write a program in 'C' language to implement a Dequeue using pointers. All operations associated with a Dequeue are to be implemented.

Code:

```
/*to implemnet a dequ using linked list using pointer **/
#include<stdio.h>
#include<conio.h>
struct node
{
int data;
struct node *link;};
void addqatend(struct node**,struct node**,int);
void addqatbeg(struct node**,struct node**,int);
delqatbeg(struct node**,struct node**);
delqatend(struct node**,struct node**);
main()
{
struct node*front,*rear;
int item;
front=rear=NULL;
addqatend(&front,&rear,11);
addqatend(&front,&rear,81);
addqatend(&front,&rear,16);
addqatend(&front,&rear,45);
```

```
clrscr();
q_display(front);
printf("\nnumber of elements in the que=%d",count(front));
printf("\nitems taken from q\n");
item=delqatbeg(&front,&rear);
printf("%d ",item);
printf("\nafter deletion\n");
q_display(front);
getch();
}
/*adds a new element at end **/
void addqatend(struct node **f,struct node **r,int item)
{
struct node *q;
q=(struct node*)malloc(sizeof(struct node));
q->data=item;
q->link=NULL;
/**if q empty**/
if(*f==NULL)
*f=q;
else
(*r)->link=q;
*r=q;
}
/*add at begin**/
void addqatbeg(struct node** f,struct node** r,int item)
{
struct node *q;
int t;
q=(struct node*)malloc(sizeof(struct node));
q->data=item;
q->link=NULL;
/**if q empty**/
if(*f==NULL)
*f=*r=q;
else
q->link=*f;
*r=*f;
*f=q;
}
/*remove from front ***/
delqatbeg(struct node** f,struct node** r)
{
struct node *q; int item;
if(*f==NULL)
printf("q empty");
else
q=*f;item=q->data;
*f=q->link; free(q);
/*if q becom empty after delet*/
if(*f==NULL)
*r=NULL;
return item;
}
/*remove from rear end ***/
delqatend(struct node** f,struct node** r)
{
struct node *q,*rleft,*temp; int item;
temp=*f;
if(*r==NULL)
printf("q empty");
else
/*traverse q to find the prevous element adrs*/
while(temp!=*r)
{rleft=temp; temp=temp->link;}
```

```
/*delete the node*/
q=*r;item=q->data;
free(q);
*r=rleft;
(*r)->link=NULL;
/*if q becom empty after delet*/
if(*r==NULL)
*f=NULL;
return item;
}
/*to display**/
q_display(struct node *q)
{
printf("\nfront->");
while(q!=NULL)
{
if(q->link==NULL)
printf("<-rear");
printf("%2d ",q->data);
q=q->link;
}
}
/*count nodes**/
count(struct node *q)
{
int c=0;
while(q!=NULL)
{
q=q->link; c++;}
return c;
}
```

Output:

```
front->11 81 16 <-rear45
number of elements in the que=4
item taken from q
11
after deletion
front->81 16 <-rear45
```

Ex 16: Write a program in 'C' language to reverse the elements of a queue.

Code:

```
/*reverse of que using arrays*/
#include<stdio.h>
#include<conio.h>
int num[10], rear=0,front=0;
add()
{
int n;
printf("Enter numbers(5 only)\n");
if(rear<5)
{
printf("num="); scanf("%d",&n);
num[rear]=n; rear++;add();} else{rear=rear-1; return;}
}
display(front)
{
if(front==rear){
printf("q empty"); return;}
printf("The numbers in reverse order are:\n");
while(front<=rear){
printf("\n%d",num[rear]); rear--;}
}
main()
{ int f;
```

```
clrscr();
add();
display();
getch();
}
```

Output:

```
Enter numbers(5 only)
num=1
num=2
num=3
num=4
num=5
The numbers in reverse order are:
5
4
3
2
1
```

Ex 17: Write a program in 'C' language to implement a queue using two stacks.

Code:

```
#include<stdio.h>
int stak1[10], stak2[10], n, top,top1;
/*add elements to que*/
void add()
{
while(top>0)
{
scanf("%d",&n);
stak1[top]=n;
top++;
}
while(top<10)
{
stak2[top1]=stak1[top];
top1++; top--;
}
}
/*delete elements from que*/
void del()
{
int n;
while(top1>0)
n=stak2[top1];
top1--;
}
/*display elements*/
void display()
{
int i=top1;
while(i>0)
{
printf("\n%d",stak2[i]);
i++;
}
}
main()
{
printf("\nEnter 10 numbers\n");
add();
display();
del();
printf("Elements in the que after deleting first ele\n");
display();
```

```
del();
printf("\nElements after deleting second ele\n");
display();
getch();
}
```

Output:

```
Enter 10 numbers to the stuk
1 2 3 4 5 6 7 8 9 0
Elements in the que:
1 2 3 4 5 6 7 8 9 0
Elements in the que after deleting first element
2 3 4 5 6 7 8 9 0
Elements after deleting 2nd element
3 4 5 6 7 8 0
```

Session 9 : Searching and Sorting

Ex 30: Write a program in 'C' language to implement linear search using pointers.

Code:

```
/*Write a program for linear searching*/
# include<stdio.h>
main()
{
int arr[20],n,i,item;
clrscr();
printf("How many elements you want to enter in the array : ");
scanf("%d",&n);
for(i=0; i < n;i++)
{
printf("Enter element %d : ",i+1);
scanf("%d", &arr[i]);
}
printf("Enter the element to be searched : ");
scanf("%d",&item);
for(i=0;i < n;i++)
{
if(item == arr[i])
{
printf("%d found at position %d\n",item,i+1);
break;
}
}/*End of for*/
if(i == n)
printf("Item %d not found in array\n",item);
getch();
}
```

Output:

```
How many elements you want to enter in the array : 5
Enter element 1 : 45
Enter element 2 : 98
Enter element 3 : 75
Enter element 4 : 86
Enter element 5 : 42
Enter the element to be searched : 75
75 found at position 3
```

Ex 31: Write a program in 'C' language to implement binary search using pointers.

Code:

```
/*binary search using pointers*/
# include<stdio.h>
# include<conio.h>
void main()
{
int search(int *,int,int,int,int *);
```

```
int arr[]={0,1,2,3,4,5,7,12,53,31,78,87,65,45,100,200};
int i,j,n=15,temp,num,pos;
char ans;
clrscr();
printf("Do  u want to enter values  to array automaticaly y/n:");
scanf("%c",&ans);
if(ans=='n')
{
printf("Enter number of elts, max is 15 :");
scanf("%d",&n);
printf("Enter %d elements...\n",n);
for(i=0;i<n;i++)
scanf("%d",&arr[i]);
}
for(i=0;i<n-1;i++)
for(j=i;j<n;j++)
if(arr[i]< arr[j])
{
temp=arr[j];
arr[j]=arr[i];
arr[i]=temp;
}
printf("\nEntered array after sorting is...\n");
for(i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
sear: printf("\nEnter the number to be searched:");
scanf("%d",&num);
if(search(arr,0,n-1,num,&pos))
printf("Entered number %d found at position %d\n",num,pos+1);
else
printf("Entered number %d not found \n",num);
printf("\Search again y/n :");
scanf(" %c",&ans);
if(ans=='y')goto sear ;
}
int search(int *arr,int spos,int epos,int num,int *pos)
{
int mid;
if(spos > epos)
{
*pos=-1;
return(0);
}
mid=(epos+spos)/2;
if(*(arr+mid)==num)
{
*pos=mid;
return(1);
}
if(*(arr+mid)> num)
{ return( search(arr,mid+1,epos,num,pos) ); }
if(*(arr+mid) < num)
{ return( search(arr,spos,mid-1,num,pos) ); }
 }
```

Output:

```
Do u want to enter values to array automaticaly y/n:y
Entered array after sorting is...
100 87 78 65 53 45 31 12 7 5 4 3 2 1 0
Enter the number to be searched:5
Entered number 5 found at position 10
Search again y/n :n
```

Ex 32: Write a program in 'C' language to implement Quick sort using pointers.

Code:

```
/****QUICK SORT ***********/
#include<stdio.h>
#include<conio.h>
int n;
qsort(int b[],int left, int right)
{
int i,j,p,tmp,finished,k;
if(right>left)
{
i=left;
j=right;
p=b[left];
printf("the partitioning element is :%d\n",p);
finished=0;
while(!finished)
{   do
{  ++i;  }
while((b[i]<=p)  && (i<=right));
while((b[j]>=p)  && (j>left))
--j;
if(j<1)
finished=1;
else
{
printf("swapping  %d & %d\n",b[i],b[j]);
tmp=b[i];
b[i]=b[j];
b[j]=tmp;
for(k=0;k<n;++k)
printf("%5d",b[k]);
printf("\n");
}  }
printf("\nI is gretaer than J,so b[left] and\n");
printf("b[j] are swapped.swapping  %d,%d\n",b[left],b[j]);
tmp=b[i];
b[i]=b[j]; b[j]=tmp;
for(k=0;k<n;++k)
printf("%5d",b[k]);
printf("\n");
}
qsort(b,left,j-1);
qsort(b,i,right);
return;
}
main()
{
int a[100],i,l,r;
clrscr();
printf("Number  of elements\n");
scanf("%d",&n);
printf("Enter elements\n");
for(i=0;i<n;++i)
scanf("%d",&a[i]);
for(i=0;i<n;++i)
printf("  %d",a[i]);
l=0; r=n-1;
qsort(a,l,r);
printf("result\n");
for(i=0;i<n;++i)
printf("  %d",&a[i]);
getch();
}
```

Output:

```
Number of elements
5
Enter elements
2 4 1 69 41
2 4 1 69 41the partitioning element is :2
swapping 4 & 1
2 1 4 69 41
swapping 4 & 1
2 4 1 69 41
I is gretaer than J,so b[left] and
b[j] are swapped.swapping  2,2
69 4 1 2 41
```

Ex 34: Write a program in 'C' language to implement 2-way Merge sort using pointers.

Code:

```
/* Program of sorting using merge sort through recursion*/
#include<stdio.h>
#define MAX 20
int array[MAX];
main()
{
int i,n;
clrscr();
printf("Enter the number of elements : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter element %d : ",i+1);
scanf("%d",&array[i]);
}
printf("Unsorted list is :\n");
for( i = 0 ; i<n ; i++)
printf("%d ", array[i]);
merge_sort( 0, n-1);
printf("\nSorted list is :\n");
for( i = 0 ; i<n ; i++)
printf("%d ", array[i]);
printf("\n");
getch();
}/*End of main()*/
merge_sort( int low, int high )
{ int mid;
if( low != high )
{
mid = (low+high)/2;
merge_sort( low , mid );
merge_sort( mid+1, high );
merge( low, mid, high );
}
}/*End of merge_sort*/
merge( int low, int mid, int high )
{
int temp[MAX];
int i = low;
int j = mid +1 ;
int k = low ;
while( (i <= mid) && (j <=high) )
{
if(array[i] <= array[j])
temp[k++] = array[i++] ;
else
temp[k++] = array[j++] ;
}/*End of while*/
while( i <= mid )
temp[k++]=array[i++];
```

```
while( j <= high )
temp[k++]=array[j++];
for(i= low; i <= high ; i++)
array[i]=temp[i];
}  /*End of merge()*/
```

Output:

```
Enter the number of elements : 5
Enter element 1 : 88
Enter element 2 : 956
Enter element 3 : 785
Enter element 4 : 456
Enter element 5 : 754
Unsorted list is :
88 956 785 456 754
Sorted list is :
88 456 754 785 956
```

Ex 35: Write a program in 'C' language to implement Bubble sort using pointers.

Code:

```
/* Program of sorting using bubble sort */
#include <stdio.h>
#define MAX 20
main()
{
int arr[MAX],i,j,k,temp,n,xchanges;
clrscr();
printf("Enter the number of elements : ");
scanf("%d",&n);
for (i = 0; i < n; i++)
{
printf("Enter element %d : ",i+1);
scanf("%d",&arr[i]);
}
printf("Unsorted list is :\n");
for (i = 0; i < n; i++)
printf("%d ", arr[i]);
printf("\n");
/* Bubble sort*/
for (i = 0; i < n-1 ; i++)
{
xchanges=0;
for (j = 0; j <n-1-i; j++)
{  if (arr[j] > arr[j+1])
        {
        temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
        xchanges++;
        }
 } /*End of inner for loop*/
if(xchanges==0) /*If list is sorted*/
break;
printf("After Pass %d elements are : ",i+1);
for (k = 0; k < n; k++)
printf("%d ", arr[k]);
printf("\n");
}/*End of outer for loop*/
printf("Sorted list is :\n");
for (i = 0; i < n; i++)
printf("%d ", arr[i]);
printf("\n");
getch();
}   /*End of main()*/
```

Output:

```
Enter the number of elements : 4
Enter element 1 : 45
Enter element 2 : 88
Enter element 3 : 75
Enter element 4 : 6
Unsorted list is : -
45 88 75 6
After Pass 1 elements are : 45 75 6 88
After Pass 2 elements are : 45 6 75 88
After Pass 3 elements are : 6 45 75 88
Sorted list is : -
6 45 75 88
```

Ex 36: Write a program in 'C' language to implement Topological sort using pointers.

Code:

```
/* Program for topological sorting */
#include<stdio.h>
#define MAX 20
int n,adj[MAX][MAX];
int front=-1,rear=-1,queue[MAX];
main()
{
int i,j=0,k;
int topsort[MAX],indeg[MAX];
clrscr();
create_graph();
printf("The adjacency matrix is :\n");
display();
/*Find the indegree of each node*/
for(i=1;i<=n;i++)
{
indeg[i]=indegree(i);
if( indeg[i]==0 )
insert_queue(i);
}
while(front<=rear)  /*Loop till queue is not empty */
{
k=delete_queue();
topsort[j++]=k; /*Add node k to topsort array*/
/*Delete all edges going fron node k */
for(i=1;i<=n;i++)
{
if( adj[k][i]==1 )
{
adj[k][i]=0;
indeg[i]=indeg[i]-1;
if(indeg[i]==0)
insert_queue(i);
}
}/*End of for*/
}/*End of while*/
printf("Nodes after topological sorting are :\n");
for(i=0;i<j;i++)
printf( "%d ",topsort[i] );
printf("\n");
getch();
}/*End of main()*/
create_graph()
{
int i,max_edges,origin,destin;
printf("Enter number of vertices : ");
scanf("%d",&n);
max_edges=n*(n-1);
```

```
for(i=1;i<=max_edges;i++)
{
printf("Enter edge %d(0 0 to quit): ",i);
scanf("%d %d",&origin,&destin);
if((origin==0) && (destin==0))
break;
if( origin > n || destin > n || origin<=0 || destin<=0)
{ printf("Invalid edge!\n"); i--; }
else
adj[origin][destin]=1;
}/*End of for*/
}/*End of create_graph()*/
display()
{ int i,j;
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
printf("%3d",adj[i][j]);
printf("\n");
} } /*End of display()*/
insert_queue(int node)
{
if (rear==MAX-1)
printf("Queue Overflow\n");
else
{
if (front==-1) /*If queue is initially empty */
front=0;
rear=rear+1;
queue[rear] = node ;
}
}/*End of insert_queue()*/
delete_queue()
{
int del_item;
if (front == -1 || front > rear)
{
printf("Queue Underflow\n");
return ;
} else
{
del_item=queue[front];
front=front+1;
return del_item;
} }/*End of delete_queue() */
int indegree(int node)
{
int i,in_deg=0;
for(i=1;i<=n;i++)
if( adj[i][node] == 1 )
in_deg++;
return in_deg; }/*End of indegree() */
```

Output:

```
Enter number of vertices : 3
Enter edge 1(0 0 to quit): 1 3
Enter edge 2(0 0 to quit): 1 2
Enter edge 3(0 0 to quit): 3 2
Enter edge 4(0 0 to quit): 0 0
The adjacency matrix is :
0 1 1
0 0 0
0 1 0
Nodes after topological sorting are :
1 3 2
```