1. Implement a Program in C for converting an Infix expression to Postfix Expression.

```
#define SIZE 10
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
char stack[SIZE];
int top=-1;
void infixtopostfix(char *,char *);
void push(char symbol)
{
  stack[++top]=symbol;
}
char pop()
{
  return(stack[top--]);
}
int priority(char op)
{
  switch(op)
  {
    case '#':
                return 0;
    case '(':
                return 1;
    case '+':
    case '-':
                return 2;
    case '*':
    case '/':
                return 3;
   }
}
void infixtopostfix(char *infix,char *postfix)
 {
  char symbol,brace;
  int i=0,k=0;
 push('#');
 while((symbol=infix[i++])!='\0')
 {
  if(isalnum(symbol))
     postfix[k++]=symbol;
  else if(symbol=='(')
     push(symbol);
  else if(symbol==')')
   {
     while(stack[top]!='(')
         postfix[k++]=pop();
      brace=pop();
    }
    else
```

```
    {
      while(priority(symbol)<=priority(stack[top]))
            postfix[k++]=pop();
      push(symbol);
    }
   }
   while(stack[top]!='#')
       postfix[k++]=pop();
   postfix[k]='\0';
  }
void main()
{
  char infix[50],postfix[50];
  clrscr();
  printf(" Enter  the infix expression \n");
  scanf("%s",infix);
  infixtopostfix(infix,postfix);
  printf("\n postfix expression is : %s",postfix);
  getch();
}
```

Output:

```
Enter  the infix expression
(A+B)*C-(D-E)
postfix expression is : AB+C*DE- -
```

2. Excute a program in c to evaluate to valid postfix expression using stack.

```
#include<stdio.h>
#include<math.h>
#include<string.h>
double compute(char symbol,double op1,double op2)
{
switch(symbol)
{
case '+':return op1+op2;
case '-':return op1-op2;
case '*':return op1*op2;
case '/':return op1/op2;
case '$':
case '^':return pow(op1,op2);
}
return (0);
}
void main()
{
double s[20];
double res,op1,op2;
int top;
int i;
char postfix[20],symbol;
```

```
printf("enter the postfix expression\n");
gets(postfix);
top=-1;
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\n",res);
getch();
}
```

Output:

```
enter the postfix expression:623*+5-
the result is:7.00
```

3. design, develop, and excute a program in c to simulate the working of queue of integers using an array.Provide the folowing operations:
a.Insert b.Delete c.Display

```
#include<stdio.h>
#include<conio.h>
#define MAX 3
int Q[MAX];
int front=-1;
int rear=-1;
void enqueue(int item)
{
if(rear==MAX-1)
{
printf("\n **queue is overflow**\n");
return;
}
if(front==-1)
front++;
Q[++rear]=item;
}
void dequeue()
{
int item;
if(front==-1)
{
```

```
printf("**Queue id unferflow**\n");
return;
}
item=Q[front];
if(front==rear)
{
front=rear=-1;
printf("**empty**\n");
}
else
front++;
printf("\n deleted item is:%d\n",item);
}
void display()
{
int temp;
if(front==-1)
{
printf("\n**queue is empty...**\n");
return;
}
for(temp=front;temp<=rear;temp++)
printf("%d",Q[temp]);
}
void main()
{
int choice,item;
clrscr();
printf("\n queue implementation");
while(1)
{
printf("\n1.insert\n 2.delete\n 3.display\n 4.exit");
printf("\n enter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
  printf("\n enter the element to be inserted:");
  scanf("%d",&item);
  enqueue(item);
  break;
case 2:
   dequeue();
   break;
case 3:
   printf("\n the contents of the queue is:\n");
   display();
   break;
default:
exit(0);
```

```
}
}
getch();
}
```

Output:

```
queue implementation
1.insert
2. delete
3.display
4.exit
enter your choice:1
enter the element to be inserted:10
1.insert
2.delete
3.display
4.exit
enter your choice:1
enter the elements to be inserted:20
1.insert
2.delete
3.display
4.exit
enter your choice:1
enter the elements to be inserted:30
1.insert
2.delete
3.display
4.exit
enter your choice:1
enter the elements to be inserted:40
1.insert
2.delete
3.display
4.exit
enter your choice:2
enter the elements to be inserted:10
1.insert
2.delete
3.display
4.exit
enter your choice:2
enter the elements to be inserted:20
1.insert
2.delete
3.display
4.exit
enter your choice:3
enter the elements to be inserted:30 40
```

4.write a c program to simulate the working of singly linked link providing the following operations.

a. display & insert b. delete from the begining/end c.delete a given element

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<stdio.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("no memeory inheap");
exit(0);
}
return x;
}
NODE insert(int item,NODE last)
{
NODE temp=getnode();
temp->info=item;
if(last==NULL)
last=temp;
else
temp->link=last->link;
last->link=temp;
return last;
}
void display(NODE last)
{
NODE temp;
if(last==NULL)
{
printf("no elements to display\n");
return;
}
temp=last->link;
printf("the contents of list\n");
while(temp!=last)
{
printf("%d  ",temp->info);
temp=temp->link;
}
```

```
printf("%d",temp->info);
}
NODE del_item(int key,NODE last)
{
NODE prev,cur;
if(last==NULL)
{
printf("list is empty");
return last;
}
if(last==last->link&&key==last->info)
{
printf("\n deleted item is %d",last->info);
free(last);
return NULL;
}
prev=last;
cur=last->link;
while(cur!=last&key!=cur->info)
{
prev=cur;
cur=cur->link;
}
if(key==cur->info)
{
prev->link=cur->link;
printf("\n deleted item is %d",cur->info);
if(cur==last)
last=prev;
free(cur);
}
else
printf("\n key not found\n");
return last;
}
NODE delete_front(NODE last)
{
NODE temp,first;
if(last==NULL)
{
printf("empty list\n");
return NULL;
}
if(last->link==last)
{
printf("deleted elements is %d\n",last->info);
free(last);
return NULL;
}
first=last->link;
```

```
last->link=first->link;
printf("item deleted is %d\n",first->info);
free(first);
return last;
}
NODE delete_rear(NODE last)
{
NODE prev;
if(last==NULL)
{
printf("empty list");
return NULL;
}
if(last->link==last)
{
printf("the item deleted is %d\n",last->info);
free(last);
return NULL;
}
prev=last->link;
while(prev->link!=last)
{
prev=prev->link;
}
prev->link=last->link;
printf("the item deleted is %d\n",last->info);
free(last);
return prev;
}
void main()
{
int opt,item,key;
NODE last=NULL;
clrscr();
for(;;)
{
printf("\n***singly linked list***\n");
printf("\n1.insert\n2.delete_front\n3.delete_rear\n4.delete a given element\n5.display\n6.exit\n");
printf("enter your opt");
scanf("%d",&opt);
switch(opt)
{
case 1:printf("\n enter item\n");
scanf("%d",&item);
last=insert(item,last);
break;
case 2:last=delete_front(last);
break;
case 3:last=delete_rear(last);
break;
```

```
case 4:printf("\n enter key elemnt to be deleted:");
scanf("%d",&key);
last=del_item(key,last);
break;
case 5:display(last);
break;
default:
exit(0);
}
}
}
```

Output:

```
***Singly Linked Link***
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:1
enter the item:10
***singly linked list***
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:1
enter the item:20
***singly linked list***
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:1
enter the item:30
***singly linked list***
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:1
enter the item:40
***singly linked list***
```

```
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:1
enter the item:50
***singly linked list***
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:2
enter the item:50
***singly linked list***
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:3
enter the item:10
***singly linked list***
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:4
enter key element to be deleted:30
deleted item is:30
***singly linked list***
1.insert
2.delete_front
3.delete_rear
4.delete a given element
5.display
6.exit
enter the opt:5
enter the item:20 40
```

5. Write a C program to Implement the following searching techniques
a. Linear Search b. Binary Search.

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

```
int a[50],n,key,i,j;
void linear();
void binary();
void main()
{
int ch;
clrscr();
printf("\enter the length of array:");
scanf("%d",&n);
printf("\n enter the array elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nenter the key element:");
scanf("%d",&key);
while(1)
{
printf("\n1.linear search\n2.binary search\n3.exit\n");
printf("\n enter your choice");
scanf("%d",&ch);
switch(ch)
{
case 1:linear();
break;
case 2:binary();
break;
case 3:return;
default:printf("\n wrong choice");
}
}
}
void linear()
{
i=0;
while(a[i]!=key)
{
i++;
}
if(i<n)
printf("the %d elements is found at the location %d:",key,i+1);
else
printf("the %d element is not found in the list:",key);
}
void binary()
{
int temp,first=0,mid,last=n-1;
for(i=0;i<=n-2;i++)
{
for(j=0;j<=n-2;j++)
{
if(a[j+1]<a[j])
```

```
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp'
}
}
}
printf("\n the sorted list is:\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
mid=(first+last)/2;
while(first<=last)
{
if(a[mid]==key)
first=mid+1;
elseif(a[mid]==key)
{
printf("\n%d found at the location %d\n:,key,mid+1);
break;
}
else last=mid-1;
mid=(first+last)/2;
}
if(first>last)
printf("the key element %d is not in the list\n:",key);
}
```

Output:

```
enter the length of array:5
enter the array elements:
10
50
30
20
40
enter the key element:20
1. linear search
2. binary search
3. exit
enter your choice:1
the 20 element is found at the location 4
enter your choice:2
the sorted list is: 10 20 30 40 50
 20 found at  the location 2
```

6.write a c program to implement the following sorting techniques.
a.bubble sort b.selection sort

```
#include<stdio.h>
#include<conio.h>
int a[50],n,i,j;
```

```
void bubble();
void selection();
void main()
{
int ch;
clrscr();
printf("\n enter the length of an array");
scanf("%d",&n);
printf("\n enter the array elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
while(1)
{
printf("\n1.bubble sort\n2.selection sort\n3.exit");
printf("\n enter your choice");
scanf("%d",&ch);
switch(ch)
{
case 1:bubble();
      break;
case 2:selection();
      break;
case 3:return;
default:printf("\n wrong choice");
}
}
}
void bubble()
{
int temp;
printf("\n before sorting");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
for(i=0;i<=n-2;i++)
{
for(j=0;j<=n-2-i;j++)
{
if(a[j+1]<a[j])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("\n after sorting ascending order\n");
for(i=0;i<n;i++)printf("%d\t",a[i]);
}
void selection()
{
```

```
int min,temp;
printf("\n before sorting");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
for(i=n-1;i>0;i--)
{
min=0;
for(j=1;j<=i;j++)
{
if(a[j]<a[min])
{
min=j;
}
}
temp=a[i];
a[i]=a[min];
a[min]=temp;
}
printf("\n after sorting in descending order\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
}
```

Output:

```
enter the length of the array: 5
enter the array elements: 10 40 50 30 20
1.bubble sort
2.selection sort
3.exit
enter your choice:1
before sorting:10 40 50 30 20
after sorting:
        ascending order: 10 20 30 40 50
1.bubble sort
2.selection sort
3.exit
enter your choice:2
before sorting: 10 20 30 40 50
after sorting:
        descending order:50 40 30 20 10
```

7. Find Minimum Cost Spanning Tree of a given undirected graph usingKruskal's algorithm (C programming)

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
```

```
int uni(int,int);
void main()
{
        clrscr();
        printf("\n\n\t Implementation of Kruskal's algorithm\n\n");
        printf("\n Enter the number of vertices\n"); scanf("%d",&n);
        printf("\n Enter the cost adjacencyy matrix\n");
        for(i=1;i<=n;i++)
        {

                for(j=1;j<=n;j++)
                {
                        scanf("%d",&cost[i][j]);
                        if(cost[i][j]==0)
                        cost[i][j]=999;
                }
}
printf("\n The edges of minimum cost spanning tree are\n\n");
while(ne<n)
{
        for(i=1,min=999;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        if(cost[i][j]<min)
                        {
                              min=cost[i][j];
                              a=u=i;
                             b=v=j;
                        }
                }
        }
       u=find(u);
       v=find(v);
       if(uni(u,v))
                {
                        printf("\n %d edge
                        (%d,%d)=%d\n",ne++,a,b,min);mincost+=min;
                }
                cost[a][b]=cost[b][a]=999;
        }
        printf("\n\t Minimum cost =%d\n",mincost);
        getch();
}

int find(int i)
{
        while(parent[i])
        i=parent[i];
```

```
        return i;
}
int uni(int i,int j)
{
        if(i!=j)
        {
                parent[j]=i;
                return 1;
        }
        return 0;
}
```

Output:

```
Enter the number of vertices:
4
Enter the cost adjacency matrix
0 1 5 2
1 0 999 999
5 999 0 2
2 999 2 0
The edges of minimum cost spanning tree are
1 edge(1,2)=1
2 edge(1,4)=2
3 edge(3,4)=2
Minimum cost=5
```

8. From a given vertex in a weightaged connected graph find shortest path to other vertices using Dijkstra's algorithm.

```
#include<stdio.h>
#include<conio.h>
int min(int a,int b)
{
        return((a<b)?a:b);
}
void dijkstra(int n,int src,int cost[10][10],int d[],int s[])
{
int i,j,u,v,small;
for(i=1;i<=n;i++)
{
        s[i]=0;
        d[i]=cost[src][i];
        }
        s[src]=1;
        for(j=1;j<=n;j++)
        {
                small=999;
                u=0;
                for(i=1;i<=n;i++)
                {
```

```
                if(d[i]<small && s[i]==0)
                {
                        small=d[i];
                        u=i;
                }
        }
        s[u]=1;
        for(v=1;v<=n;v++)
        {
                if(s[v]==0)
        {
                d[v]=min(d[v],d[u]+cost[u][v]);
        }
    }
}
}
        int main()
        {
        int src,i,j,n,cost[10][10],d[10],s[10];
        printf("Enter the number of vertex\n");
        scanf("%d",&n);
        printf("Enter the adjecency matrix");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        scanf("%d",&cost[i][j]);
        printf("Enter the source vertex\n");
        scanf("%d",&src);
        dijkstra(n,src,cost,d,s);
        for(i=1;i<=n;i++)
        printf("shortest path form %d to %d is %d\n",src,i,d[i]);
        getch();
}
```

Output:

```
Enter the number of vertex
5
Enter the adjacency matrix
0 4 2 999 999
4 0 999 12 8
2 999 0 6 9
999 12 6 0 9
999 8 999 9 0
Enter the source vertex
1
Shortest path from 1 to 1 is 0
Shortest path from 1 to 2 is 4
Shortest path from 1 to 3 is 2
Shortest path from 1 to 4 is 8
Shortest path from 1 to 5 is 11
```