Program 1 :Write a C program to create a sequential file with at least 5 records, each record having the structure shown below.
a. To display all records in the file.
b. To search for a specific record based on the USN. In case the record is not found, suitable message should be displayed. Both the options in this case must be demonstrated.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct mark
{
intusn;
char name[25];
int mark1,mark2,mark3;

};

void display(FILE *fp)
{
struct mark m;
inti;
for(i=0;i<=5;i++)
   {
printf("The USN,NAME,mark1,mark1,mark1 of student %d is \n",i+1);
fscanf(fp,"%d %s %d %d %d",&m.usn,m.name,&m.mark1,&m.mark2,&m.mark3);
printf("%d %s %d %d %d\n",m.usn,m.name,m.mark1,m.mark2,m.mark3);
   }
}
void search(FILE *fp,intusn)
{
struct mark m;
inti;
for(i=0;i<=5;i++)
   {
fscanf(fp,"%d %s %d %d %d",&m.usn,m.name,&m.mark1,&m.mark2,&m.mark3);
if(usn==m.usn)
   {
printf("The USN,NAME,mark1,mark2,mark3 of student is\n");
printf("%d %s %d %d %d\n",m.usn,m.name,m.mark1,m.mark2,m.mark3);
return;
   }

   }
printf("\n The student with the usn is not found\n");
}

void main()
{
struct mark m;
inti,ch,usn,n;
   FILE *fp;
clrscr();
fp=fopen("aa.doc","w");
for(i=0;i<5;i++)
   {
```

```
printf("Enter the USN,NAME,mark1,mark2,mark3 of student %d \n",i+1);
scanf("%d%s%d%d%d",&m.usn,m.name,&m.mark1,&m.mark2,&m.mark3);
fprintf(fp,"%d %s %d %d %d\n",m.usn,m.name,m.mark1,m.mark2,m.mark3);
   }
fclose(fp);
for(;;)
   {
printf("Enter choice\n1.DISPLAY\n2.SEARCH\n");
scanf("%d",&ch);
switch(ch)
      {
case 1:fp=fopen("aa.doc","r");
display(fp);
fclose(fp);
break;
case 2:printf("\n Enter the USN to be searched\n");
scanf("%d",&usn);
fp=fopen("aa.doc","r");
search(fp,usn);
fclose(fp);
break;
default: exit(0);

      }
   }

}
```

Sample Output :

```
Enter the usn name marks1 marks2 marks3 of student 1 is
12 arun 10 20 30
Enter the usn name marks1 marks2 marks3 of student 2 is
13 nikhil 30 50 60
Enter the usn name marks1 marks2 marks3 of student 3 is
14 rahul 40 30 40
Enter the usn name marks1 marks2 marks3 of student 4 is
15 mehul 12 33 45
Enter the usn name marks1 marks2 marks3 of student 5 is
16 satish 34 22 11
Enter 1 for display 2 for search based on usn, others to exit
1
The usn marks1 marks2 marks3 of student 1 is
12 arun 10 20 30
The usn marks1 marks2 marks3 of student 2 is
13 nikhil 30 50 60
The usn marks1 marks2 marks3 of student 3 is
14 rahul 40 30 40
The usn marks1 marks2 marks3 of student 4 is
15 mehul 12 33 45
The usn marks1 marks2 marks3 of student 5 is
16 satish 34 22 11
Enter 1 for display 2 for search based on usn, others to exit
```

Program 2 : Design, develop and execute a C program to check if a string is a palindrome or not.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
clrscr();
char string1[20];
inti,length;
int flag=0;
printf("Enter a string\n");
scanf("%[^\n]s",string1);
length=strlen(string1);
for(i=0;i<length/2;i++)
    {
if(string1[i]!=string1[length-i-1])
flag=1;
break;
    }
if(flag)
printf("\n%s is not a palindrome string",string1);
else
printf("\n%s is a palindrom string");
getch();
}
```

Sample Output:

```
Enter a string
Malayam
Malayam is a palindrome string

Enter a string
Abacus
Abacus is not a palindrome string
```

Program 3: Design, develop and execute a program in C to convert a given valid parenthesized infix arithmetic expression to postfix expression and then print both the expressions. The expression consists of single character operands and the binary operators +, - , * , /

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
int top=-1;
char stack[40];
void push(char x)
{
stack[++top]=x;
}
int pop()
{
return stack[top--];
}
int prior(char x)
{
int p;
if(x=='('||x=='#')
   p=1;
if(x=='+'||x=='-')
   p=2;
if(x=='*'||x=='/')
   p=3;
if(x=='^'||x=='$')
   p=4;
return p;
}
void main()
{
char infix[30],postfix[30];
inti,j=0;
clrscr();
printf("Enter the infix expressin:\n");
gets(infix);
push('#');
for(i=0;i<strlen(infix);i++)
    {
if(isalnum(infix[i]))
postfix[j++]=infix[i];
else if(infix[i]=='(')
push(infix[i]);
else if(infix[i]==')')
   {
while(stack[top]!='(')
postfix[j++]=pop();
pop();
   }
else
   {
while(prior(stack[top])==prior(infix[i]))
postfix[j++]=pop();
push(infix[i]);
   }
    }
while(stack[top]!='#')
postfix[j++]=pop();
postfix[j]='\0';
printf("The poostfixexpressin is:\n");
puts(postfix);
getch();
}
```

Sample Output:

1) Enter an infix expression:
   (a+b)*c-(d/e)

   Postfix expression is:ab+c*de/-

2) Enter an infix expression:
   (1+2)-(a/b)*c/(p+q+r)

   Postfix expression is:12+ab/c*pq+r+/-

Program 4: Design, develop, and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations:

- **a. Insert**
- **b. Delete**
- **c. Display**

```
#include<stdio.h>
#include<conio.h>
#define MAXSIZE 5
int f=-1,r=-1,q[MAXSIZE];
void insert()
{
int item;
if(r==MAXSIZE-1)
printf("OVERFLOW!!!");
else
   {
if(f==-1)
          f=0;
printf("Enter the items to be inserted:\n");
scanf("%d",&item);
q[++r]=item;

   }
}
void delete()
{
if(f==-1)
printf("UNDERFLOW!!!");
else if(f>r)
   {
      f=r=-1;
printf("UNDERFLOW!!!");
   }
else
printf("item deleted=%d",q[f++]);
}
void display()
{
inti;
if(f==-1)
printf("UNDERFLOW!!!");
else
   {
printf("The elements are: \n");
for(i=f;i<=r;i++)
printf("%d ",q[i]);
   }
}
void main()
{
intch;
for(;;)
   {
printf("\n1:INSERT\n2:DELETE\n3:DISPLAY\n4:EXIT\n");
scanf("%d",&ch);
switch(ch)
    {
case 1:printf("INSERTION\n");
insert();
break;
case 2:printf("DELETION\n");
delete();
break;
case 3:printf("DISPLAY\n");
display();
break;
default  :exit(0);
    }
    }
}
```

Sample Output:

```
1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
enter element 10
1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
enter element20
1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
enter element30

1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
enter element40
1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
OVERFLOW

1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
3
10 20 30 40
1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
2
element deleted is 10

.....Q operations.....
1.add   2.delete       3.display       4.exit
enter your choice->2
element deleted is 20

1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
2
element deleted is 30

1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
2
element deleted is 40

1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
```

```
2
UNDERFLOW
1.INSERT 2:DELETE 3:DISPLAY 4:EXIT
4
```

Program 5 :Design, develop and execute a program in C to implement linked list to insert and delete an element from the list.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *link;
};
typedefstruct node* NODE;

NODE getnode()
{
   NODE x;
   x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{  printf("out of memory\n");
exit(0);
   }
return x;
}
voidfreenode(NODE x)
{
free(x);
}
NODE insert_front(intitem,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("Link is empty\n");
return first;
   }
temp=first;
temp=temp->link;
printf("The deleted item is %d \n",first->info);
freenode(first);
return temp;
}
NODE insert_rear(intitem,NODE first)
{
   NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
```

```
return temp;
cur=first;
while(cur->link!=0)
   {
cur=cur->link;
   }
cur->link=temp;
return first;
}
NODE delete_rear(NODE first)
{
   NODE cur=first,prev=NULL;
if(cur==NULL)
   {
printf("Link is empty\n");
return first;
   }
while(cur->link!=NULL)
   {
prev=cur;
cur=cur->link;
   }
freenode(cur);
prev->link=NULL;
return first;
}
void display(NODE first)
{  NODE temp=first;
if(first==NULL)
   {
printf("NO NODES IN THE LIST!!\n");
return first;
   }
else
   {
while(temp!=NULL)
      {
printf("%d ",temp->info);
temp=temp->link;
      }
   }
}
void main()
{
   NODE first=NULL;
intchoice,item;
clrscr();
for(;;)
   {
printf("\n1.INSERT FRONT\n2.INSERT REAR\n3.DELETE FRONT\n4.DELETE REAR\n5.DISPLAY\n");
scanf("%d",&choice);
switch(choice)
      {
case 1:printf("Enter the item\n");
scanf("%d",&item);
first=insert_front(item,first);
break;
case 2:printf("Enter the item\n");
scanf("%d",&item);
first=insert_rear(item,first);
break;
case 3:first=delete_front(first);
break;
case 4:first=delete_rear(first);
break;
case 5:display(first);
break;
default:exit(0);
        }
    }
}
```

Sample Output:

```
1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
1
20

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
1
30

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
1
40

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
5
40 30 20

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
2
10

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
2
5

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
5
40 30 20 10 5

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
3

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
5
30 20 10 5

1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
4
1.INSERT FRONT 2.INSERT REAR 3.DELETE FRONT 4.DELETE REAR 5.DISPLAY
```

```
5
30 20 10
```

Program 6: Design, develop, and execute a program in C to read a sparse matrix of integer values and to search the sparse matrix for an element specified by the user. Print the result of the search appropriately. Use the triple <row, column, value> to represent an element in the sparse matrix.

```
#include<stdio.h>
#include<conio.h>
#define SROW 50
#define MROW 20
#define MCOL 20
int main()
{
int mat[MROW][MCOL],sparse[SROW][3];
inti,j,nzero=0,mr,mc,sr,s,elem;
printf("Enter number of rows:\n");
scanf("%d",&mr);
printf("Enter number of column:\n");
scanf("%d",&mc);
printf("Enter the matrix\n");
for(i=1;i<=mr;i++)
    {
for(j=1;j<=mc;j++)
        {
scanf("%d",&mat[i][j]);
if(mat[i][j]!=0)
            {
nzero++;
            }
        }
    }
sr=nzero+1;
sparse[1][1]=mr;
sparse[1][2]=mc;
sparse[1][3]=nzero;
    s=2;
for(i=1;i<=mr;i++)
    {
for(j=1;j<=mc;j++)
        {   if(mat[i][j]!=0)
            {
sparse[s][1]=i;
sparse[s][2]=j;
sparse[s][3]=mat[i][j];
s++;
            }
        }
    }
printf("\nSparse matrix is \n");
for(i=1;i<=sr;i++)
    {
for(j=1;j<=3;j++)
        {
printf("%d ",sparse[i][j]);
        }
printf("\n");
    }
printf("Enter the element to be searched \n");
scanf("%d",&elem);
for(i=2;i<sr;i++)
    {
if(sparse[i][3]==elem)
        {printf("Element found at (row,col)=(%d,%d)",sparse[i][1],sparse[i][2]);
getch();
return 1;
        }
    }
printf("Element not found");
getch();
return 0;

}
```

Sample Output:

```
Enter number of rows: 3
Enter number of columns: 3
Enter the Matrix:
0 0 1
0 0 2
0 0 3
Sparse matrix is:
   3    3    3
   1    3    1
   2    3    2
   3    3    3
Enter the element to be searched: 3
Element found at (row, col) : (3, 3)
```

Program 7: Design, develop, and execute a program in C to create a max heap of integers by accepting one element at a time and by inserting it immediately in to the heap. Use the array representation for the heap. Display the array at the end of insertion phase.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAXSIZE 10
int insertion(intitem,int a[],int n)
{
intc,p;
if(n==MAXSIZE)
   {
printf("HEAP IS FULL!!!\n");
return;
   }
   c=n;
   p=(c-1)/2;
while(c!=0&&item>a[p])
   {
a[c]=a[p];
         c=p;
         p=(c-1)/2;
   }
a[c]=item;
return n+1;
}
void display(int a[],int n)
{
inti;
if(n==0)
   {
printf("HEAP IS EMPTY!!!\n");
return;
   }
printf("The array elements are \n");
for(i=0;i<n;i++)
printf("%d ",a[i]);
}
void main()
{
int a[MAXSIZE],n=0,ch,item;
clrscr();
for(;;)
   {
printf("\n1.INSERT\n2.DISPLAY\nEXIT\n");
scanf("%d",&ch);
switch(ch)
       {
case 1:printf("Enter the element");
scanf("%d",&item);
                n=insertion(item,a,n);
break;
case 2:display(a,n);
break;
default:exit(0);
```

```
        }
    }
}
```

Sample Output:

```
1. Insert     2: display          3:exit
enter the choice
1
enter the item to be inserted
2
1. Insert     2: display          3:exit
enter the choice
1
enter the item to be inserted
3
1. Insert     2: display          3:exit
enter the choice
1
enter the item to be inserted
5
1. Insert     2: display          3:exit
enter the choice
1
enter the item to be inserted
4
1. Insert     2: display          3:exit
enter the choice
1
enter the item to be inserted
6
1. Insert     2: display          3:exit
enter the choice
2
the array elements are
6
5
3
2
4
```

Program 8:Design, develop, and execute a program in C to implement a doubly linked list where each node consists of integers. The program should support the following operations:

i. **Create a doubly linked list by adding each node at the front.**
ii. **Insert a new node to the left of the node whose key value is read as an input.**
iii. **Delete the node of a given data if it is found, otherwise display appropriate message**
iv. **Display the contents of the list.**

(Note: Only either (a, b and d) or (a, c and d) may be asked in the examination)

```
#include<stdio.h>
#include<conio.h>
struct node
{
int info;
struct node *llink;
struct node *rlink;
};
typedefstruct node* NODE;

NODE getnode()
{
   NODE x;
   x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
   {
printf("Out of Memory\n");
exit(0);
   }
return x;
}
voidfreenode(NODE x)
{
free(x);
}
NODE insert_front(intitem,NODE first)
{
   NODE temp;
temp=getnode();
temp->info=item;
if(first==NULL)
   {
temp->rlink=NULL;
temp->llink=NULL;
return temp;
   }
first->llink=temp;
temp->llink=NULL;
temp->rlink=first;
return temp;
}
NODE insert_keyvalue(NODE first,intkey,int item)
{
   NODE prev,cur,temp;
if(first==NULL)
   {
printf("List is Empty\n");
return NULL;
```

```
   }
prev=NULL;
cur=first;
while(cur->rlink!=NULL&&cur->info!=key)
   {
prev=cur;
cur=cur->rlink;
   }
if(cur->info==key)
    {
temp=getnode();
temp->info=item;
temp->llink=prev;
temp->rlink=cur;
cur->llink=temp;
prev->rlink=temp;
    }
else
    {
printf("Specified key not found\n");
    }
return first;
}
NODE delete_key(intkey,NODE first)
{
   NODE prev,cur,next=NULL;
prev=NULL;
if(first==NULL)
   {
printf("List is empty");
return NULL;
   }
cur=first;
while(cur->rlink!=NULL&&cur->info!=key)
   {
prev=cur;
cur=cur->rlink;
   }
if(cur->info==key)
   {
printf("DELETED!\n");
prev->rlink=cur->rlink;
cur->rlink->llink=prev;
freenode(cur);
   }
else
   {
printf("Specified item not found\n");
   }
return first;
}
void display(NODE first)
{NODE cur;
if(first==NULL)
 {
printf("List is Empty\n");
```

```
return;
 }
cur=first;
while(cur->rlink!=NULL)
 {
printf("%d ",cur->info);
cur=cur->rlink;
 }
printf("%d ",cur->info);
}
void main()
{
   NODE first;
intchoice,item,key;
for(;;)
   {
printf("\1.insert front        2.insert before       3.delete all occurrence    4.display list  5.exit\n");
scanf("%d",&choice);
switch(choice)
       {
case 1:printf("Enter item to insert\n");
scanf("%d",&item);
first=insert_front(item,first);
break;
case 2:printf("Enter key element\n");
scanf("%d",&key);
printf("Enter item to insert\n");
scanf("%d",&item);
first=insert_keyvalue(first,key,item);
break;
case 3:printf("Enter element to be deleted\n");
scanf("%d",&key);
first=delete_key(key,first);
break;
case 4:display(first);
break;
default:exit(0);

       }
   }
}
```

Sample Output:

```
..... double link list.......
1.insert front       2.insert before       3.delete all occurrence    4.display list  5.exit
enter your choice 1
enter the item 10

1.insert front       2.insert before       3.delete all occurrence    4.display list  5.exit
enter your choice 1
enter the item 20

1.insert front       2.insert before       3.delete all occurrence    4.display list  5.exit
enter your choice 4
.... list status...
```

```
20  10

1.insert front      2.insert before     3.delete all occurrence    4.display list  5.exit
enter your choice 2
enter the key 20
enter the item 15

1.insert front      2.insert before     3.delete all occurrence    4.display list  5.exit
enter your choice 4
.... list status...
15  20  10

1.insert front      2.insert before     3.delete all occurrence    4.display list  5.exit
enter your choice 3
enter item to be deleted:10
item 10 deleted...

1.insert front      2.insert before     3.delete all occurrence    4.display list  5.exit
enter your choice  3
enter item to be deleted:15
item 15 deleted...

1.insert front      2.insert before     3.delete all occurrence    4.display list  5.exit
enter your choice  3
enter item to be deleted:20
item 20 deleted...

1.insert front      2.insert before     3.delete all occurrence    4.display list  5.exit
enter your choice   3
list is empty
```

Program 9:Design, develop, and execute a Quick Sort algorithm in C.

```
#include<stdio.h>
#include<conio.h>
int partition(int a[],intlow,int high)
 {
intp,i,j,temp;
     p=a[low];
i=low+1;
     j=high;
while(1)
     {
while(a[i]<=p&&i<high)
i++;
while(a[j]>p&&j>=low)
j--;
if(i<j)
        {
temp=a[i];
a[i]=a[j];
a[j]=temp;

        }

temp=a[j];
a[j]=a[low];
a[low]=temp;
return j;
 }
 }
void quicksort(int a[],intlow,int high)
 {
int s;
if(low<high)
     {
         s=partition(a,low,high);
quicksort(a,low,s-1);
quicksort(a,s+1,high);
     }
 }
void main()
 {
int a[10],i,n;
clrscr();
printf(" Enter the size of array");
scanf("%d",&n);
printf("\nEnter the array elements\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
quicksort(a,0,n-1);
printf("\nSorted elements are:\n");
for(i=0;i<n;i++)
printf("%d ",a[i]);
getch();
 }
```

Sample Output:

```
Enter the size of array
5
Enter the array elements
40 10 50 28 30
Sorted elements are:
10 28 30 40 50
```

Program 10: Design, develop and execute a program in C to create a Binary Tree and to perform inorder, preorder and postorder traversals.

```
#include<stdio.h>
#include<stdlib.h>
struct node {
int data;
struct node *leftChild;
struct node *rightChild;
};
struct node *root = NULL;
void insert(int data) {
struct node *tempNode = (struct node*) malloc(sizeof(struct node));
struct node *current;
struct node *parent;
tempNode->data = data;
tempNode->leftChild = NULL;
tempNode->rightChild = NULL;
//if tree is empty
if(root == NULL) {
root = tempNode;
} else {
current = root;
parent = NULL;
while(1) {
parent = current;
//go to left of the tree
if(data < parent->data) {
current = current->leftChild;
//insert to the left
if(current == NULL) {
parent->leftChild = tempNode;
return;
}
} //go to right of the tree
else {
current = current->rightChild;
//insert to the right
if(current == NULL) {
parent->rightChild = tempNode;

return;
}
}
}
}
}
voidpre_order_traversal(struct node* root) {
if(root != NULL) {
printf(“%d “,root->data);
pre_order_traversal(root-&gt;leftChild);
pre_order_traversal(root-&gt;rightChild);
}
}
voidinorder_traversal(struct node* root) {
if(root != NULL) {
```

```
inorder_traversal(root-&gt;leftChild);
printf("%d &",root->data);
inorder_traversal(root->rightChild);
}
}
voidpost_order_traversal(struct node* root) {
if(root != NULL) {
post_order_traversal(root->leftChild);
post_order_traversal(root->rightChild);
printf("%d", root->data);
}
}
int main() {
inti;
int array[7] = { 27, 14, 35, 10, 19, 31, 42 };
for(i = 0; i&lt; 7; i++)
insert(array[i]);
printf("\nPreorder traversal: ");
pre_order_traversal(root);
printf("\nInorder traversal:");
inorder_traversal(root);
printf("\nPost order traversal: ");
post_order_traversal(root);
return 0;
}
```

Sample Output:

```
Preorder traversal: 27 14 10 19 35 31 42
Inorder traversal: 10 14 19 27 31 35 42
Post order traversal: 10 19 14 31 42 35 27
```