Data Structures Lab

1. Write a C program that uses functions to perform the following:
a) Create a singly linked list of integers.
b) Delete a given integer from the above linked list.
c) Display the contents of the above list after deletion.

Aim : To Create a Singly Linked List of Integers and Delete a given integer from the linked list.
and Display the contents of the List after deletion.

Program :

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

struct node
{
        int data;
        struct node *next;
};

typedef struct node NODE;

NODE *head=NULL;

void create();
void delete();
void display();

void create()
{
        int val ,op;
        printf("Enter the value for the node data:\n");
        scanf("%d",&val);
        NODE *newnode=(NODE *)malloc(sizeof(NODE));
        newnode -> data =val;
        newnode -> next =NULL;

        if(head==NULL)
        {
                head=newnode;
        }
        else
        {
                int ele;
                printf("Enter \n 1.insert at start \n 2.insert at end \n 3. insert at middle \n ");
                scanf("%d",&op);

                switch(op)
                {               //at starting
                        case 1:
                                {
                                        newnode -> next = head;
```

```
                    head=newnode;

                }
                break;

            case 2:
                //at end
                {
                    NODE *ptr;
                    ptr=head;
                    while(ptr ->next != NULL)
                        ptr=ptr -> next;
                    ptr->next=newnode;

                }
                break;
            case 3:
                {
                    printf("Enter the element to be inserted after which of the
element :")      ;
                    scanf("%d",&ele);
                    NODE *ptr=head;
                    while(ptr->data != ele)
                    {
                        ptr=ptr->next;
                    }
                    newnode->next=ptr->next;
                    ptr->next=newnode;
                    break;
                }
        }
    }
}

void delete()
{
    int ele;
    if(head==NULL)
    {
        printf("No Elements to Delete \n");
    }
    else
    {
        display();
        printf("\n Enter the elements to be delete :");
        scanf("%d",&ele);
        NODE *ptr=head;
        NODE *oldptr=NULL;

        while(ptr -> data != ele)
        {
            oldptr=ptr;
```

```
                        ptr=ptr ->next;
                }
                if(ptr==head)
                {
                        //head deletion
                        head=ptr -> next;

                }
                else if(ptr ->next == NULL)
                {
                        //last node deletion
                        oldptr -> next = NULL;
                }
                else
                {
                        //middle node deletion
                        oldptr -> next=ptr -> next;
                }

                printf("The %d element is sucessfully deleted \n",ptr->data);
                free(ptr);
                display();
        }
}


void display()

{
        NODE *ptr=head;
        if(ptr==NULL)
        {
                printf("NO Elements present in the linked list \n");
        }
        else
        {
                while(ptr != NULL)
                {
                        printf("%d ---> ",ptr -> data);
                        ptr=ptr -> next;
                }
        }
}

void main()

{
        int opt;
        while(1)
        {
                printf("\n Enter \n 1.create \n 2.delete \n 3.exit \n");
                scanf("%d",&opt);
                switch(opt)
                {
                        case 1:
                                create();
                                break;

                        case 2:
                                delete();
                                break;
                        case 3:
                                exit(0);
                        default:
                                printf("Enter the correct values for option \n");



                }
        }
}
```

Output :

```
 Enter
 1.create
 2.delete
 3.exit
1
Enter the value for the node data:
10

 Enter
 1.create
 2.delete
 3.exit
1
Enter the value for the node data:
20
Enter
 1.insert at start
 2.insert at end
 3. insert at middle
 2

 Enter
 1.create
 2.delete
 3.exit
1
Enter the value for the node data:
30
Enter
 1.insert at start
 2.insert at end
 3. insert at middle
 2

 Enter
 1.create
 2.delete
 3.exit
2
10 ---> 20 ---> 30 --->
 Enter the elements to be delete :20
The 20 element is sucessfully deleted
10 ---> 30 --->
 Enter
 1.create
 2.delete
 3.exit
 3
```

2. Write a C program that uses functions to perform the following:
a) Create a doubly linked list of integers.
b) Delete a given integer from the above doubly linked list.
c) Display the contents of the above list after deletion.

Aim : program that uses functions to perform Create a doubly linked list of integers and Delete a given integer from the doubly linked list and Display the contents of the list after deletion.

Program :

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        struct node *prev;
        int data;
        struct node *next;
};
typedef struct node NODE;
NODE *head = NULL;
void create();
void display();
void delete();


void create()
{
        int a;
        printf("Enter the value:");
        scanf("%d",&a);
        NODE *newnode = (NODE*)malloc(sizeof(NODE));
```

```
    newnode->data=a;
    newnode->next=NULL;
    newnode->prev=NULL;
    if(head == NULL)
    {
        head=newnode;
    }
    else
    {
        NODE *ptr=head;
        int opt;
        printf(" \n INSERT IN \n 1.starting \n 2.middle\n3.ending\n");
        scanf("%d",&opt);
        switch(opt)
        {
            case 1:
            {
                newnode->next=head;
                head->prev=newnode;
                newnode->prev=NULL;
                head=newnode;
                break;
            }
            case 2:
            {
                int m;
                printf("\n Enter before value to insert after that:");
                scanf("%d",&m);
                while(ptr->data != m)
                {
                    ptr=ptr->next;
                }
                newnode->next=ptr->next;
                newnode->prev=ptr;
                ptr->next=newnode;
                ptr->next->prev=newnode;
                break;
            }
            case 3:
            {
                while(ptr->next != NULL)
                    ptr=ptr->next;
                newnode->next=NULL;
                newnode->prev=ptr;
                ptr->next=newnode;
                break;
            }
        }
    }
}
void display()
{
    NODE *ptr = head;
    if(ptr == NULL)
        printf("no elements in list");
    while(ptr != NULL)
    {
        printf(" %d--->",ptr->data);
        ptr=ptr->next;
    }

}
void delete()
{
    NODE *ptr = head;
    int s;
    if(head == NULL)
        printf("no elements in list\n");
    else
    {
        display();
        printf("\n Enter the number to delete:");
        scanf("%d",&s);
        NODE *ptr=head;
        NODE *oldnode=NULL;
        while(ptr->data != s)
        {
            oldnode=ptr;
            ptr=ptr->next;
        }
        if(ptr == head)
        {
            head=ptr->next;
            ptr->next->prev=NULL;   // head node
        }
        else if(ptr->next == NULL)   // last node
        {
            oldnode->next=NULL; /// head node
        }
        else
        {
            oldnode->next=ptr->next;
            ptr->next->prev=oldnode;
        }
        free(ptr);
        display();

    }


}
void main()
{
    int choose;
```

```
	while(1)
	{
		printf("\nEnter \n1.create\n2.delete\n3.exit\n");
		scanf("%d",&choose);
		switch(choose)
		{
			case 1:create();break;
			case 2:delete();break;
			case 3:exit(0);break;
			default:printf("choose a valuble number:\n");
		}
	}
}
```

Output:

```
Enter
1.create
2.delete
3.exit
1
Enter the value:10

Enter
1.create
2.delete
3.exit
1
Enter the value:20

 INSERT IN
 1.starting
 2.middle
3.ending
3

Enter
1.create
2.delete
3.exit
1
Enter the value:30

 INSERT IN
 1.starting
 2.middle
3.ending
3

Enter
1.create
2.delete
3.exit
2
```

```
 10---> 20---> 30--->
 Enter the number to delete:30
 10---> 20--->
Enter
1.create
2.delete
3.exit
3
```

3. Write a C program implement the Stack ADT using Arrays

Aim : Program implement the Stack ADT using Arrays

Program :

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

#define N 5

int stack[N];

int top=-1;

void push();
int pop();
void display();

void push()
{
	if(top==N-1)
	{
		printf("stack overflow \n");
	}
	else
	{
		int a;
		printf("Enter the element into the stack :");
		scanf("%d",&a);
		top++;
		stack[top]=a;
	}
}

int pop()
{
	int val;
	if(top==-1)
	{
		return (-1);
	}
	else
	{
        val=stack[top];
        top--;
        return(val);
    }
}

void display()
{
    if(top==-1)
    {
        printf("No Elements Present In the Stack \n");
    }
    else
    {
        int i ;
        printf("The Elements Present In The Stack Are....\n ");
        for(i=top;i>=0;i--)
        {
            printf("%d \n",stack[i]);
        }
    }
}

void main()
{
    int op,d;
    while(op)
    {
        printf("\n Enter \n 1.For push \n 2.For pop \n 3.For display \n 4.For exit \n");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                {
                    push();
                    break;
                }
            case 2:
                {
                    d=pop();
                    if(d==-1)
                        printf("stack underflow\n");
                    else
                        printf(" %d element is deleted ",d);
                    break;
                }
            case 3:
                {
                    display();
                    break;
                }
            case 4:
```

```
                                                {
                                                        exit(0);
                                                }
                        }
            }
}
```

Output :

```
 Enter
 1.For push
 2.For pop
 3.For display
 4.For exit
1
Enter the element into the stack :10

 Enter
 1.For push
 2.For pop
 3.For display
 4.For exit
1
Enter the element into the stack :20

 Enter
 1.For push
 2.For pop
 3.For display
 4.For exit
1
Enter the element into the stack :30

 Enter
 1.For push
 2.For pop
 3.For display
 4.For exit
3
The Elements Present In The Stack Are....
 30
20
10

 Enter
 1.For push
 2.For pop
 3.For display
 4.For exit
2
 30 element is deleted
 Enter
 1.For push
 2.For pop
 3.For display
 4.For exit
3
The Elements Present In The Stack Are....
 20
10

 Enter
 1.For push
 2.For pop
 3.For display
 4.For exit
4
```

4. Write a C program implement the Stack ADT using Linked List.

Aim : Program implement the Stack ADT using Linked List

Program :

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	int data;
	struct node *next;
};

typedef struct node NODE;

NODE *head=NULL;

void push();
void pop();
void display();

void push()
{
	int a;
	printf("Enter the element into the stack :");
	scanf("%d",&a);
	NODE *newnode;
	newnode=(NODE *)malloc(sizeof(NODE));
	newnode->data=a;
	newnode->next=NULL;

	if(head==NULL)
	{
		head=newnode;
	}
	else
```

```
    {
        newnode->next=head;
        head=newnode;
    }
}
void  pop()
{
    NODE *ptr=head;
    if(head==NULL)
    {
        printf("\n_____stack underflow______\n");
    }
    else
    {
        printf("%d is the deleted elment",ptr->data);
        head=ptr->next;
        free(ptr);
    }
}
void display()
{
    NODE *ptr=head;
    if(head==NULL)
    {
        printf("No Elements Present In The Stack");
    }
    else
    {
        while(ptr != NULL)
        {
            printf("%d\n",ptr->data);
            ptr=ptr->next;
        }
    }
}

void main()
{
    int op;
    while(1)
    {
        printf("\n Enter \n 1.For push \n 2. For pop \n 3.For display \n 4. For exit \n");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                {
                    push();
                    break;
                }
            case 2:
                {
                    pop();
                    break;
                }
            case 3:
                {
                    display();
                    break;
                }
            case 4:
                {
                    exit(0);
                }
        }
    }
}
```

Output :

```
 Enter
 1.For push
 2. For pop
 3.For display
 4. For exit
1
Enter the element into the stack :10

 Enter
 1.For push
 2. For pop
 3.For display
 4. For exit
1
Enter the element into the stack :20

 Enter
 1.For push
 2. For pop
 3.For display
 4. For exit
1
Enter the element into the stack :30

 Enter
 1.For push
 2. For pop
 3.For display
 4. For exit
3
30
20
10
```

```
 Enter
 1.For push
 2. For pop
 3.For display
 4. For exit
2
30 is the deleted elment
 Enter
 1.For push
 2. For pop
 3.For display
 4. For exit
4
```

5. Write a C program that uses stack operations to convert a given infix expression into its postfix equivalent.

Aim : Program that uses stack operations to convert a given infix expression into its postfix equivalent.

Program :

```
#include<stdio.h>
#include<string.h>
void push(char);
char pop();
int isoperator(char);
int precedence(char);
char infix[50];
char postfix[50];
int i;
char stack[40];
int top=-1;
char pop()
{
        char opr=stack[top];
        top--;
        return opr;
}
void push(char op)
{
        stack[++top]=op;
}
int precedence(char op)
{
        if(op=='^')
                return 6;
        else if(op=='*')
                return 5;
        else if(op=='%')
                return 4;
        else if(op=='/')
        return 3;
    else if(op=='+')
        return 2;
    else if(op=='-')
        return 1;
    else
        return 0;
}
int isoperator(char op)
{
    if(op=='^'|| op=='%'||op=='/'||op=='*'||op=='+'||op=='-')
        return 1;
    else
        return 0;
}
void evalpostfix(char *infix)
{
    int j,cp;
    char temp[50];
    for(j=0;j<strlen(infix);j++)
    {
        temp[j]=infix[j];
    }
    temp[j]=')';
    j++;
    temp[j]='\0';
    push('(');
    for(j=0;j<strlen(temp);j++)
    {
        if((temp[j]>=65&&temp[j]<=90)||(temp[j]>=97&&temp[j]<=122))
            postfix[i++]=temp[j];
        else if(temp[j]=='(')
            push(temp[j]);
        else if(isoperator(temp[j])==1)
        {
            cp=precedence(temp[j]);
            while(precedence(stack[top])>=cp)
                postfix[i++]=pop();
            push(temp[j]);
        }
        else if(temp[j]==')')
        {
            while(stack[top]!='(')
                postfix[i++]=pop();
            char last=pop();
        }
    }
    printf("%s\n",postfix);
}
void main()
{
    printf("Enter the infix expression:");
```

```
        scanf("%s",infix);
        evalpostfix(infix);
}
```

Output :

```
Enter the infix expression:A+B*C
ABC*+
```

6. Write a C program that evaluates a postfix expression.

Aim : Program that evaluates a postfix expression.

Program :

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int stack[50];
int top=-1;
void push(int v)
{
        stack[++top]=v;

}
int pop()
{
        int c=stack[top];
        top--;
        return c;

}
void eval_postfix(char *postfix)
{
        int j, val,op1,op2,t1;
        int size=strlen(postfix);
        for(j=0;j<size;j++)
        {
                if((postfix[j]>=65&&postfix[j]<=90)||(postfix[j]>=97&&postfix[j]<=122))
                {
                        printf("Enter the value of %c ",postfix[j]);
                        scanf("%d",&val);
                        push(val);
                }
                else // operator
                {
                        op1=pop();
                        op2=pop();
                        if(postfix[j]=='^')
                                t1=op2^op1;
                        else if(postfix[j]=='%')
                t1=op2%op1;
            else if(postfix[j]=='/')
                t1=op2/op1;
            else if(postfix[j]=='*')
                t1=op2*op1;
            else if(postfix[j]=='+')
                t1=op2+op1;
            else
                t1=op2-op1;
            push(t1);
        }
    }
    printf("The value of the expression is %d ",pop());
}
void main()
{
    char postfix[50];
    printf("Enter the postfix expression :");
    scanf("%s",postfix);
    eval_postfix(postfix);
}
```

Output :

```
Enter the postfix expression :abc*+
Enter the value of a 12
Enter the value of b 2
Enter the value of c 3
The value of the expression is 18
```

7. Write C program to implement queue ADT using array and doubly linked list.

Aim : a) Program to implement queue ADT using array

Program :

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
int queue[MAX];
int front=-1,rear=-1;
void enqueue();
void dequeue();
void display();

void enqueue()
{
    if(rear==MAX-1)
        printf("\nQUEUE OVERFLOW\n");
    else
    {
        if(rear==-1 && front==-1)
            front=rear=0;
        else
        {
            rear=rear+1;
        }
        printf("ENter the element to be inserted\n");
        scanf("%d",&queue[rear]);
    }
}
void dequeue()
{
    if(front==-1 || front>rear)
        printf("\nUnderflow\n");
    else
    {
        printf("The element deleted is %d\n",queue[front]);
        front=front+1;
    }


}
void display()
{
    int i;
    if(front==-1||front>rear)
        printf("\nUnder flow\n");
    else
    {
        for(i=front;i<=rear;i++)
        printf("%d\n",queue[i]);
    }
}
void main()
{
    int op;
    while(1)
    {
        printf("\n1.enqueue\n2.dequeue\n3.display\n4.exit\n");
        scanf("%d",&op);
        switch(op)
        {
            case 1: enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:display();
                break;
            case 4:
                exit(0);
        }
    }
}
```

Output :

```
1.enqueue
2.dequeue
3.display
4.exit
1
ENter the element to be inserted
10

1.enqueue
2.dequeue
3.display
4.exit
1
ENter the element to be inserted
20

1.enqueue
2.dequeue
3.display
4.exit
1
ENter the element to be inserted
30

1.enqueue
2.dequeue
3.display
4.exit
3
10
20
30

1.enqueue
2.dequeue
3.display
4.exit
2
The element deleted is 10

1.enqueue
2.dequeue
3.display
4.exit
4
```

Aim : b) Program to implement queue ADT using doubly linked list.

Program :

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

```
#include<stdlib.h>

struct node
{
        struct node *prev;
        int data;
        struct node *next;
};

typedef struct node NODE;

NODE *head=NULL;

void enqueue();
void dequeue();
void display();

void enqueue()
{
        int val;
        printf("Enter the element :");
        scanf("%d",&val);
        NODE *rear=(NODE*)malloc(sizeof(NODE));
        rear->data=val;
        rear->next=NULL;
        rear->prev=NULL;
        if(head==NULL)
        {
                head=rear;
        }
        else
        {
                NODE *ptr=head;
                while(ptr->next != NULL)
                {
                        ptr=ptr->next;
                }
                rear->next=NULL;
                rear->prev=ptr;
                ptr->next=rear;
        }
}

void dequeue()
{
        NODE *ptr=head;
        NODE *oldnode=NULL;
        if(head==NULL)
        {
                printf("No Elements Present :");
        }
        else
```

```
    {

        if(ptr==head)
        {
            head=ptr->next;
            ptr->next->prev=NULL;
        }
    }
    free(ptr);
    display();
}

void display()
{
    NODE *ptr = head;
    if(ptr == NULL)
        printf("no elements in list");
    while(ptr != NULL)
    {
        printf("%d \t",ptr->data);
        ptr=ptr->next;
    }
}

void main()
{
    while(1)
    {
        int opt;
        printf("\nEnter\n\t 1 for enqueue \n\t 2 for dequeue \n\t 3 for display \n\t 4 for exit \
n");
        scanf("%d",&opt);
        switch(opt)
        {
            case 1:enqueue();
                break;
            case 2:dequeue();
                break;
            case 3:display();
                break;
            case 4:exit(0);
        }
}
}
```

Output :

```
Enter
     1 for enqueue
     2 for dequeue
     3 for display
     4 for exit
1
Enter the element :10
```

```
Enter
        1 for enqueue
        2 for dequeue
        3 for display
        4 for exit
1
Enter the element :20

Enter
        1 for enqueue
        2 for dequeue
        3 for display
        4 for exit
1
Enter the element :30

Enter
        1 for enqueue
        2 for dequeue
        3 for display
        4 for exit
3
10      20      30
Enter
        1 for enqueue
        2 for dequeue
        3 for display
        4 for exit
2
20      30
Enter
        1 for enqueue
        2 for dequeue
        3 for display
        4 for exit
4
```

8. Write C program to implement circular queue ADT using array.

Aim : Program to implement circular queue ADT using array

Program :

```
#include<stdio.h>
#include<stdlib.h>
#define max 5
int queue[max];
int front=-1,rear=-1;
void enqueue();
void dequeue();
void display();
void main()
{
        int opt;
        while(1)
        {
                printf("1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n4.EXIT\n");
                scanf("%d",&opt);
                switch(opt)
                {
                        case 1:
                                enqueue();
                                break;
                        case 2:
                                dequeue();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                exit(0);
                        default:
                                printf("Enter correct option");
                }
        }
}

void enqueue()
{
        if((rear==max-1 &&front==0)|| (front==rear+1))
                printf("\n------------QUEUE OVER FLOW-----------\n\n");
        else
        {
                if(rear==-1 && front==-1)
                {
                        front=rear=0;
                }
                else
                {
                        if(rear==max-1)
                                rear=0;
                        else
                                rear=rear+1;
                }
                printf("Enter the element:");
                scanf("%d",&queue[rear]);
        }
}

void dequeue()
{
        if(front==-1)
```

```
			printf("\n----------QUEUE UNDER FLOW---------\n\n");
		else
		{
			printf("Deleted element is %d \n",queue[front]);
			if(front==rear)
			{
				front=-1;
				rear=-1;
			}
			else
			{
				if(front==max-1)
					front=0;
				else
					front=front+1;
			}
		}
}

void display()
{
		int i;
		if(front==-1)
			printf("\n----------NO elements in the queue----------\n\n");
		else
		{
			if(front<rear)
			{
				for(i=front;i<=rear;i++)
					printf("%d  ",queue[i]);

			}
			else
			{
				for(i=front;i<max-1;i++)
					printf("%d  ",queue[i]);
				for(i=0;i<=rear;i++)
					printf("%d  ",queue[i]);
			}
		}
}
```

Output :

```
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter the element:10
1.ENQUEUE
2.DEQUEUE
```

```
3.DISPLAY
4.EXIT
1
Enter the element:20
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter the element:30
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
1
Enter the element:40
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
3
10  20  30  40  1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
2
Deleted element is 10
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
3
20  30  40  1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
4
```

9. Write C program for implementing the following sorting methods:

a) Insertion sort   b) Merge sort

Aim : a) Program for implementing the Insertion sort

Program :

```
#include<stdio.h>
#define MAX 5

void main()
{
        int i,j=0,k,key,a[MAX];
        for(i=0;i<MAX;i++)
        {
                printf("Enter the element ");
        scanf("%d",&a[i]);
    }
    for(i=1;i<MAX;i++)
    {
        key=a[i];
        j=i-1;
        while(j>=0&&a[j]>key)
        {
            a[j+1]=a[j];
            j=j-1;
        }
        a[j+1]=key;

    }
    for(k=0;k<MAX;k++)
    {
        printf("%d\t",a[k]);
    }

}
```

Output :

```
Enter the element 20
Enter the element 40
Enter the element 60
Enter the element 90
Enter the element 70
20      40      60      70      90
```

Aim : b)Program for implementing the Merge Sort

Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
//#define size 100
void MERGE(int [],int,int,int);
void MERGE_SORT(int [],int,int);
void main()
{
    int arr[100];
    int i,n;
    printf("Enter the no of elements you want in the array:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enetr the elemnts into the array:");
        scanf("%d",&arr[i]);
    }
    MERGE_SORT(arr,0,n-1);
    for(i=0;i<n;i++)
    {
```

```
		printf("%d\t",arr[i]);
	}
}
void MERGE(int arr[],int p,int q,int r)
{
	int n1,n2,i,j,k;
	n1=q-p+1;
	n2=r-p;
	int L[n1],R[n2];
	for(i=0;i<n1;i++)
	{
		L[i]=arr[p+i];
	}
	for(j=0;j<n1;j++)
	{
		R[j]=arr[q+j+1];
	}
	i=j=0;
	k=p;
	while(i<n1 && j<n2)
	{
		if(L[i]<=R[j])
		{
			arr[k]=L[i]	;
			i++;
			k++;
		}
		else
		{
			arr[k]=R[j];
			k++;
			j++;

		}
	}
	if(i<n1)
	{
		while(i<n1)
		{
			arr[k]=L[i];
			k++;
			i++;
		}
	}
	else
	{
		while(j<n2)
		{
			arr[k]=R[j];
			j++;
			k++;
		}
```

```
        }


}
void MERGE_SORT(int arr[],int p,int r)
{
        int q;
        if(p<r)
        {
                q=floor((p+r)/2);
                MERGE_SORT(arr,p,q);
                MERGE_SORT(arr,q+1,r);
                MERGE(arr,p,q,r);
        }
}
```

Output :

```
Enter the no of elements you want in the array:5
Enetr the elemnts into the array:1
Enetr the elemnts into the array:4
Enetr the elemnts into the array:6
Enetr the elemnts into the array:7
Enetr the elemnts into the array:8
1       4       6       7       8
```

10. Write C program for implementing the following sorting methods:
a) Quick sort   b) Selection sort

Aim : a) Program for implementing the Quick sort

Program :

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

void quick_sort(int A[],int,int);
void main()
{
        int A[100];
        int i,n;
        printf("Enter the no of elements you want in the array:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enetr the elements into the array:");
                scanf("%d",&A[i]);
        }
        quick_sort(A,0,n-1);
        for(i=0;i<n;i++)
        {
                printf("%d\t",A[i]);
        }
}
```

```
int partition(int A[],int lb,int ub)
{
        int pivot,start;
        int end,temp;
        pivot=A[lb];
        start=lb;
        end=ub;
        while(start<end)
        {
                while(A[start]<=pivot)
                {
                        start++;
                }
                while(A[end]>pivot)
                {
                        end--;
                }
                if(start<end)
                {
                        temp=A[start];
                        A[start]=A[end];
                        A[end]=temp;
                }
        }
        temp=A[lb];
        A[lb]=A[end];
        A[end]=temp;
        return(end);
}
void quick_sort(int A[],int lb,int ub)
{
        int val;
        if(lb<ub)
        {
                val=partition(A,lb,ub);
                quick_sort(A,lb,val-1);
                quick_sort(A,val+1,ub);
        }
}
```

Output :

```
Enter the no of elements you want in the array:5
Enetr the elements into the array:15
Enetr the elements into the array:45
Enetr the elements into the array:67
Enetr the elements into the array:89
Enetr the elements into the array:90
15      45      67      89      90
```

Aim :b) Program for implementing the Selection Sort

Program :

```
#include<stdio.h>
#define MAX 5

void main()
{
        int min,t,i,j,A[MAX];
        for(i=0;i<MAX;i++)
        {
                printf("Enter the element: ");
                scanf("%d",&A[i]);
        }
        for(i=0;i<MAX-1;i++)
        {
                min=i;
                for(j=i+1;j<MAX;j++)
                {
                        if(A[j]<A[min])
                                min=j;

                }
                t=A[min];
                A[min]=A[i];
                A[i]=t;

        }
        for(int k=0;k<MAX;k++)
        {
                printf("%d\t",A[k]);
        }
}
```

```
Output :
Enter the element: 12
Enter the element: 34
Enter the element: 56
Enter the element: 2
Enter the element: 3
2       3       12      34      56
```

11. Write a C program that uses functions to perform the following:

   a) Create a Binary Search Tree (BST).
   b) Insert data in BST
   c) Traverse the above BST recursively in Postorder.
   d) Deletion an element BST

Aim : Program that Create a Binary Search Tree (BST) and Insert data in BST and Traverse the BST recursively in Postorder.

Program: