VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

DATA STRUCTURE LAB RECORD

Submitted by

S.K.BALAJI(1BM19CS134)

Under the Guidance of

Prof. Lohith J J
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-2020 to Jan-2021

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)

CERTIFICATE

This is to certify that the LAB RECORD carried out by **S K BALAJI (1BM19CS134)** who is the bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide   Signature of the HOD
Prof. Prof. Lohith J J   Dr. Umadevi V
Assistant Professor   Associate Prof.& Head, Dept. of CSE
BMSCE, Bengaluru   BMSCE, Bengaluru

External Viva

Name of the Examiner   Signature with date

1._____   _____

2. _____   _____

Lab Programs:

Lab Programs 1:

```c
#include <stdio.h>

#include <stdlib.h>

#define SIZE 5

int top=-1;

int stack[SIZE];

void push(int ele)

{if(top==SIZE-1)

   { printf("The stack is full\n");   }

   else{

      top++;

      stack[top]=ele;}}

int pop()

{   if(top==-1)

   {     return 0}

   else

   {  printf("Element removed is : %d\n",stack[top--]);

      return 1;

   }}

void display()

{

   if(top==-1)

      printf("The stack is empty\n");

   else

   {

      printf("The elements are\n");

      for(int i=0;i<=top;i++)

      {

         printf("%d\n",stack[i]);

      }} }
```

```c
int main()
{
  int c,d,p;
  while(c!=4)
  {
  printf("Enter command\t1-push\t2-pop\t3-Display\t4-Exit\n");
  scanf("%d",&c);
  switch(c)
  {
    case 1:printf("Enter an element\n");
        scanf("%d",&d);
        push(d);
        break;
    case 2:p=pop();
        if(p==0)
         printf("Stack is empty\n");
         else
         printf("\nElement removed succesfully\n");
         break;
    case 3:display();
        break;
    case 4:break;
    default: printf("Invalid input\n");
}}return 0;}
```

```
stackWeek1.exe
Enter command    1-push   2-pop    3-Display      4-Exit
1
Enter an element
123
Enter command    1-push   2-pop    3-Display      4-Exit
1
Enter an element
564
Enter command    1-push   2-pop    3-Display      4-Exit
1
Enter an element
232
Enter command    1-push   2-pop    3-Display      4-Exit
2
Element removed is : 232

Element removed succesfully
Enter command    1-push   2-pop    3-Display      4-Exit
3
The elements are
123
564
Enter command    1-push   2-pop    3-Display      4-Exit
432
Invalid input
Enter command    1-push   2-pop    3-Display      4-Exit
3
The elements are
123
564
Enter command    1-push   2-pop    3-Display      4-Exit
4
Press any key to continue . . .
```

Lab Programs 2:

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int F(char symbol)

{

```c
switch(symbol)
{
  case'+':
  case'-':return 2;
  case'*':
  case'/':return 4;
  case'^':
  case'$':return 5;
  case'(':return 0;
  case'#':return -1;
  default:return 8;}}
int G(char symbol)
{
switch(symbol)  {
  case'+':
  case'-':return 1;
  case'*':
  case'/':return 3;
  case'^':
  case'$':return 6;
  case'(':return 9;
  case')':return 0;
default:return 7;
  }}
int infix_postfix(char infix[],char postfix[])
{int top,i,j,d=0,f=0;
char s[30],symbol;
top=-1;
s[++top]='#';
j=0;
for(i=0;i<strlen(infix);i++)
{
```

```c
    if(infix[i]=='('){
      d++;}
      else if (infix[i]==')')
      f++;
symbol=infix[i];
while(F(s[top])>G(symbol))
{
postfix[j]=s[top--];
j++;
}
if(F(s[top])!=G(symbol))
s[++top]=symbol;
else
top--;
}

while(s[top]!='#')
{
postfix[j++]=s[top--];
}
postfix[j]='\0';
return (d+f);
}

void main()
{int a;
char infix[20];
char postfix[20];
printf("Enter the valid infix expression ");
scanf("%s",infix);
a= infix_postfix(infix , postfix );
if((strlen(postfix)+a)!=strlen(infix))
```
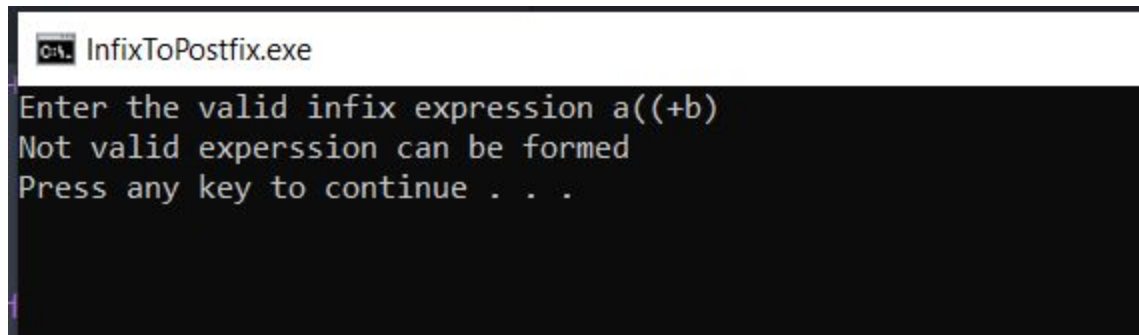
```
printf("Not valid experssion can be formed \n");

else

printf("The postfix expression is :\t%s\n",postfix);

}
```
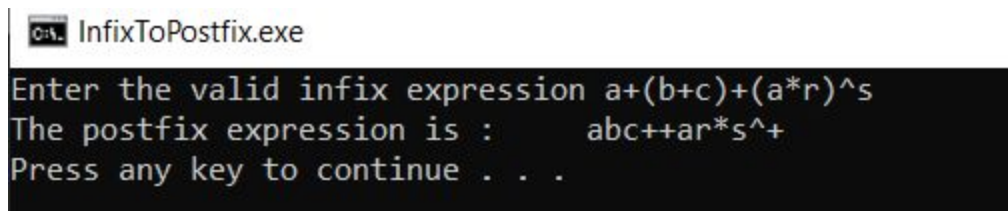

Lab Programs 3:

```
#include<stdio.h>

#include<conio.h>

#include <stdlib.h>

#define QSIZE 5

int item,front=0,rear=-1,q[10];

void insertrear(){

  if(rear ==QSIZE-1){

    printf("Queue is Overflow");

    return;

  }rear +=1;

  q[rear] = item;

}

int deletfront(){

  if(front>rear){

    front =0;
```

```c
    rear =-1;
    return -1;
  }return q[front++];
}
void display(){
  int i;
  if(front> rear){
    printf("quene is empty\n");
    return;
  }
  printf("contents of queue\n");
  for(i=front;i<=rear;i++)
  printf("%d\n",q[i]);
}
void main()
{ char ch='b';
  int choice;
  for(;;){
  printf("1.insert_rear\t2.delete_front\t3.display\t4.exit:\n");
  printf("enter choice\n");
  scanf("%d",&ch);
  switch(ch)
  {
    case 1:printf("enter the item:\t");
      scanf("%d",&item);
      insertrear();
      break;
    case 2:item =deletfront();
    if(item==-1){
      printf("Queue is UnderFlow\n");break;}
      printf("item Deleted: %d \n",item);
      break;
```

9

```
        case 3:display();
        break;
    default:exit(0);
  } }}
```


Lab Programs 4:

```
#include<stdio.h>

#include<stdlib.h>

#include<process.h>

#define que_size 3

int item,front=0,rear=-1,q[que_size],count=0;

void insertrear()

{
```

```c
        if(count==que_size)
        {
                printf("queue overflow\n");
                return;
        }
        rear=(rear+1)%que_size;
        q[rear]=item;
        count++;
}
int deletefront()
{
        if(count==0) return -1;
        item = q[front];
        front=(front+1)%que_size;
        count=count-1;
        return item;
}
void displayq()
{
        int i,f;
        if(count==0)
        {
                printf("queue is empty");
                return;
        }
        f=front;
        printf("contents of queue \n");
        for(i=0;i<count;i++)
        {
                printf("%d\n",q[f]);
                //f=(f+1)%que_size;
        }
```

```c
}
void main()
{
	int choice;
	for(;;)
	{
		printf("\n1.Insert rear \n2.Delete front \n3.Display \n4.exit \n ");
		printf("Enter the choice : ");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1:printf("Enter the item to be inserted :");
				scanf("%d",&item);
				insertrear();
				break;
			case 2:item=deletefront();
					if(item==-1)
					printf("queue is empty\n");
					else
					printf("item deleted is %d \n",item);
					break;
			case 3:displayq();
					break;
			default:exit(0);
		}
	}
}
```

```
 Enter the choice : 2
queue is empty

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 1
Enter the item to be inserted :34

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 3
contents of queue
34

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 2
item deleted is 34

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 3
queue is empty
1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 1
Enter the item to be inserted :32

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice :
```

Lab Programs 5:

```c
#include<stdio.h>

#include <conio.h>

#include<stdlib.h>

#include <process.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("memory full\n");

exit(0);

}

return x;

}

void freenode(NODE x)

{

free(x);

}

NODE insert_front(NODE first,int item)

{

NODE temp;
```

```c
temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL)

return temp;

temp->link=first;

first=temp;

return first;

}

    NODE insert_pos(int item,int pos,NODE first)

    {

    NODE temp;

    NODE prev,cur;

    int count;

    temp=getnode();

    temp->info=item;

    temp->link=NULL;

    if(first==NULL && pos==1)

    return temp;

    if(first==NULL)

    {

     printf("invalid pos\n");

     return first;

    }

    if(pos==1)

    {

    temp->link=first;

    return temp;

    }

    count=1;

    prev=NULL;

    cur=first;
```

```c
    while(cur!=NULL && count!=pos)
    {
     prev=cur;
     cur=cur->link;
     count++;
    }
    if(count==pos)
    {
    prev->link=temp;
    temp->link=cur;
    return first;
    }
    else
    printf("Invailed position ,item cannot be inserted\n");
    return first;
    }
    NODE insert_rear(NODE first,int item)
    {
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
    return temp;
    cur=first;
    while(cur->link!=NULL)
    cur=cur->link;
    cur->link=temp;
    return first;
    }
void display(NODE first)
{
```

```c
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("1:Insert_front\t2:Insert_rear\t3:Insert_AtSpecfiedLocation\t4:Display_list\t5:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 3: printf("enter the item to be inserted at loaction\n");
scanf("%d",&item);
printf("enter the position :\t");
int pos ;
scanf("%d",&pos);
```

```c
first = insert_pos(item,pos,first);

break;

case 4:display(first);

break;

default:exit(0);

break;

}

}

getch();

}
```

```
 1:Insert_front
  2:Insert_rear
 3:Insert_AtSpecfiedLocation
 4:Display_list
5:Exit
enter the choice
1
enter the item at front-end
134

 1:Insert_front
  2:Insert_rear
 3:Insert_AtSpecfiedLocation
 4:Display_list
5:Exit
enter the choice
1
enter the item at front-end
161

 1:Insert_front
  2:Insert_rear
 3:Insert_AtSpecfiedLocation
 4:Display_list
5:Exit
enter the choice
2
enter the item at rear-end
189

 1:Insert_front
  2:Insert_rear
 3:Insert_AtSpecfiedLocation
 4:Display_list
5:Exit
enter the choice
3
enter the item to be inserted at loaction
2
enter the position :    2
```

```
 1:Insert_front
  2:Insert_rear
 3:Insert_AtSpecfiedLocation
 4:Display_list
5:Exit
enter the choice
4
161
2
134
189

 1:Insert_front
  2:Insert_rear
 3:Insert_AtSpecfiedLocation
 4:Display_list
5:Exit
enter the choice
2
enter the item at rear-end
200

 1:Insert_front
  2:Insert_rear
 3:Insert_AtSpecfiedLocation
 4:Display_list
5:Exit
enter the choice
4
161
2
134
189
200

 1:Insert_front
  2:Insert_rear
 3:Insert_AtSpecfiedLocation
 4:Display_list
5:Exit
enter the choice
5
Press any key to continue . . .
```

Lab Programs 6:

```c
#include<stdio.h>

#include <conio.h>

#include<stdlib.h>

#include <process.h>

struct node

{

int info;
```

```c
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("memory full\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}

NODE delete_front(NODE first)
{
```

```c
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is= %d\n",first->info);
free(first);
return temp;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
```

```c
    printf("item deleted at rear-end is %d\n",cur->info);
    free(cur);
    prev->link=NULL;
    return first;
}
NODE delete_pos(int pos,NODE first){
  NODE prev,cur;
  int count;
  if(first==NULL)
  {
  printf("list is empty cannot delete And invalid position\n");
  return first;}
if(first->link==NULL && pos==1)
  {
  printf("item deleted is %d\n",first->info);
  free(first);
  return NULL;
  }count=1;
  prev=NULL;
  cur=first;
  while(cur!=NULL && count!=pos)
  {
  prev=cur;
  cur=cur->link;
  count++;
  }
  if(count==pos)
  {
  prev->link=cur->link;
  printf("item deleted is %d\n",cur->info);
  free(cur);
  return first;
  }
```

```c
    else
     printf("Invalid position.\n");
    return first;
    }


void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("1:Insert\t2:Delete_front\t3:Delete_rear\t4:Delete_AtSpecfiedLocation\t5:Display_list\t6:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:first=delete_front(first);
break;
```

```c
case 3:first=delete_rear(first);
break;
case 4:printf("enter the position :\t");
    int pos1 ;
    scanf("%d",&pos1);
   first = delete_pos(pos1,first);
    break;

case 5:display(first);
break;
default:exit(0);
break;
}}}
```

```
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
1
enter the item at front-end
200
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
1
enter the item at front-end
190
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
1
enter the item at front-end
80
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
1
enter the item at front-end
170
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
1
enter the item at front-end
150
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
1
enter the item at front-end
130
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
5
130
150
170
80
190
200
```

```
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
2
item deleted at front-end is= 130
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
3
item deleted at rear-end is 200
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
4
enter the position :    3
item deleted is 80
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
5
150
170
190
1:Insert        2:Delete_front  3:Delete_rear    4:Delete_AtSpecfiedLocation       5:Display_list  6:Exit
enter the choice
6
Press any key to continue . . .
```

Lab Programs 7:

#include<stdio.h>

#include <conio.h>

#include<stdlib.h>

#include <process.h>

struct node

{

int info;

26

```c
struct node *link;

};

typedef struct node *NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

{

printf("memory full\n");

exit(0);

}

return x;

}

void freenode(NODE x)

{

free(x);

}

NODE insert_front(NODE first,int item)

{

NODE temp;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL)

return temp;

temp->link=first;

first=temp;

return first;

}


NODE delete_front(NODE first)
```

```c
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is= %d\n",first->info);
free(first);
return temp;
}
NODE reverse(NODE first)
{
NODE cur,temp;
cur=NULL;
while(first!=NULL)
{
 temp=first;
 first=first->link;
 temp->link=cur;
 cur=temp;
}
return cur;
}
NODE concat(NODE first,NODE second)
{
NODE cur;
if(first==NULL)
 return second;
if(second==NULL)
```

```c
 return first;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=second;
return first;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
NODE sortList(NODE first) {
    NODE current = first, index = NULL;
    int temp;

    if(first == NULL) {
        printf("list is empty.");
      return current;
    }
    else {
      while(current != NULL) {


          index = current->link;


          while(index != NULL) {
```

```c
            if(current->info > index->info) {

                temp = current->info;

                current->info = index->info;

                index->info = temp;

            }

            index = index->link;

        }

        current = current->link;

    }

                        return current;

    }

  }

void main()

{

int item,choice,pos,n,i;

NODE first=NULL,a,b;

for(;;)

{

printf("\n1:Insert_front\t2:Delete_front\t3:reverse_list\t4:Concate\t5:Sort\t6:display_list\t7:Exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("enter the item at front-end\n");

scanf("%d",&item);

first=insert_front(first,item);

break;

case 2:first=delete_front(first);

break;

case 3: first=reverse(first); display(first);break;

 case 4:

 if(first==NULL){
```

```c
 printf("enter the no of nodes in 1:");
     scanf("%d",&n);
     a=NULL;
     for(i=0;i<n;i++)
      {
       printf("enter the item:");
       scanf("%d",&item);
       a=insert_front(a,item);
     }}else{
       a=first;
     }
      printf("enter the no of nodes in list2:");
     scanf("%d",&n);
     b=NULL;
     for(i=0;i<n;i++)
      {
       printf("enter the item:");
       scanf("%d",&item);
       b=insert_front(b,item);
      }
      a=concat(a,b);
      display(a);
     break;
case 5:sortList(first);
                display(first);
     break;
case 6:display(first);
break;
default:for(;first->link!=NULL;first=first->link)free(first);
exit(0);
break;}}}
```

```
1:Insert_front  2:Delete_front  3:reverse_list  4:Concate      5:Sort  6:display_list  7:Exit
enter the choice
1
enter the item at front-end
23

1:Insert_front  2:Delete_front  3:reverse_list  4:Concate      5:Sort  6:display_list  7:Exit
enter the choice
1
enter the item at front-end
44

1:Insert_front  2:Delete_front  3:reverse_list  4:Concate      5:Sort  6:display_list  7:Exit
enter the choice
4
enter the no of nodes in list2:2
enter the item:33
enter the item:3
44
23
3
33

1:Insert_front  2:Delete_front  3:reverse_list  4:Concate      5:Sort  6:display_list  7:Exit
enter the choice
3
33
3
23
44

1:Insert_front  2:Delete_front  3:reverse_list  4:Concate      5:Sort  6:display_list  7:Exit
enter the choice
5
3
23
33
44

1:Insert_front  2:Delete_front  3:reverse_list  4:Concate      5:Sort  6:display_list  7:Exit
enter the choice
7
Press any key to continue . . .
```

Lab Programs 8:

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<conio.h>

#include<process.h>

struct node{

struct   node *link;

 int info;

```c
};
typedef struct node *NODE;
NODE freenode(NODE x){
  free(x);
}
NODE getnode(){
  NODE x = (NODE)malloc(sizeof(struct node));
  if(x==NULL){
    printf("Memory is full\n");
    exit(0);
  }
  return x;
}
NODE insertfront(NODE first,int item){
  NODE temp =getnode();
  temp->info = item;
  temp->link = NULL;
  if(first == NULL){
    return temp;
  }
  temp->link = first;
  first = temp;
  return first;
}

NODE deletefront(NODE first){
  if(first ==NULL){
    printf("Stack is Empty\n");
    return first;
  }
  NODE temp = first;
  first = first->link;
```

```c
    printf("item POPED = %d\n",temp->info);

    freenode(temp);

    return first;

}

NODE deleterear(NODE first){

    NODE prev,curr;

    if(first == NULL){

        printf("Queue Empty\n");

        return first;

    }

    if(first->link == NULL){

        printf("item Delete at rear end is: %d\n",first->info);

        free(first);

        return NULL;

    }

    curr = first;

    prev = NULL;

    while(curr->link != NULL){

        prev = curr;

        curr = curr->link ;

    }

    prev->link  = NULL;

    printf("item delete from Queue is = %d\n",curr->info);

    freenode(curr);

    return first;

}

void display(NODE first){

NODE temp;

for(temp=first;temp!=NULL;temp=temp->link){

printf("%d\n",temp->info);

}}

int main(){
```

```c
int item,choice;
NODE first =NULL,first2 =NULL;
for(;;){
  printf("1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue 6:Display Queue  6:Exit : \n  ");
  printf("Enter The Choice: \t");
  scanf("%d",&choice);
  switch(choice){
    case 1 : printf("Enter item:\t");
         scanf("%d",&item);
        first= insertfront(first,item); break;
    case 2 :first=deletefront(first);break;
    case 3 :  if(first==NULL)
          printf("Stack empty cannot display items\n");
          else  display(first); break;
    case 4: printf("Enter item:\t");
        scanf("%d",&item);
        first2 = insertfront(first2,item);break;
    case 5: first2 = deleterear(first2);
          break;
    case 6 : if(first2 ==NULL)
        printf("Queue empty cannot display items\n");
         else display(first2);break;
    default :   exit(1);break; }}}
```

```
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      1
Enter item:      23
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      1
Enter item:      45
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      3
45
23
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      2
item POPED = 45
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      4
Enter item:      200
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      4
Enter item:      200
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      4
Enter item:      800
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      6
800
200
200
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      5
item delete from Queue is = 200
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      2
item POPED = 23
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      5
item delete from Queue is = 200
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      6
800
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      5
item Delete at rear end is: 800
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      2
Stack is Empty
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      6
Queue empty cannot display items
1:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue    6:Display Queue  6:Exit :
   Enter The Choice:      7
Press any key to continue . . .
```

Lab Programs 9:

```
#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<stdlib.h>

struct node

{

        int info;
```

```c
        struct node *llink;

         struct node *rlink;

        };

typedef struct node *NODE;

NODE getnode()

{

        NODE x;

        x=(NODE)malloc(sizeof(struct node));

        if(x==NULL)

        {

                printf("mem full\n");

                exit(0);

                }

        return x;

        }

void freenode(NODE x)

{

        free(x);

}

NODE dinsert_front(int item,NODE head)

{

NODE temp,cur;

temp=getnode();

temp->info=item;

cur=head->rlink;

head->rlink=temp;

temp->llink=head;

temp->rlink=cur;

cur->llink=temp;

return head;

}

NODE dinsert_leftpos(int item,NODE head ,int pos){
```

```c
    NODE temp,cur,perv;temp=getnode();temp->info=item;
    int i=1;
    cur=head->rlink;
    perv=NULL;
    while(i<pos && cur!=head){
     perv =cur;
    cur=cur->rlink;i++;
    }
    if(cur==head)
    {
     printf("POSITION not found\n");
     return head;
     }
    perv ->rlink=temp;
    temp->rlink=cur;
    temp->llink=perv;
    cur->llink =temp;
    return head;
}
NODE dinsert_rear(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->llink;
head->llink=temp;
temp->rlink=head;
temp->llink=cur;
cur->rlink=temp;
return head;
}
NODE ddelete_front(NODE head)
```

```c
{
NODE cur,next;
if(head->rlink==head)
{
printf("dq empty\n");
return head;
}
cur=head->rlink;
next=cur->rlink;
head->rlink=next;
next->llink=head;
printf("the node deleted is %d",cur->info);
freenode(cur);
return head;
}
NODE ddelete_rear(NODE head)
{
NODE cur,prev;
if(head->rlink==head)
{
printf("dq empty\n");
return head;
}
cur=head->llink;
prev=cur->llink;
head->llink=prev;
prev->rlink=head;
printf("the node deleted is %d",cur->info);
freenode(cur);
return head;
}
void display(NODE head)
```

```c
{
```

```c
{
NODE temp;
if(head->rlink==head)
{
printf("dq empty\n");
return;
}
printf("contents of dq\n");
temp=head->rlink;
while(temp!=head)
{
printf("%d \t",temp->info);
temp=temp->rlink;
}
printf("\n");
}
void main()
{
NODE head,last;
int item,pos, choice;
head=getnode();
head->rlink=head;
head->llink=head;

for(;;)
{
        printf("\n1:insert front\t2:insert rear\t3:delete front\t4:delete
rear\t5:display\t6:left-side-insert\t7:exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
```

```
                case 1: printf("enter the item at front end\n");

                        scanf("%d",&item);

                        last=dinsert_front(item,head);

                        break;

                case 2: printf("enter the item at rear end\n");

                        scanf("%d",&item);

                        last=dinsert_rear(item,head);

                        break;

                case 3:last=ddelete_front(head);

                        break;

                case 4: last=ddelete_rear(head);

                        break;

                case 5: display(head);

                        break;

        case 6:  printf("enter the item at left side pos to entered\n");

                        scanf("%d",&item);

  printf("POSITION\t");

                        scanf("%d",&pos);

                        last=dinsert_leftpos(item,head,pos);

                        break;

                default:exit(0);

                }}}
```

```
1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
1
enter the item at front end
100

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
2
enter the item at rear end
190

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
1
enter the item at front end
200

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
5
contents of dq
200     100     190

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
6
enter the item at left side pos to entered
2
POSITION        2

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
5
contents of dq
200     2       100     190

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
2
enter the item at rear end
1202
```

```
1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
5
contents of dq
200     2       100     190

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
2
enter the item at rear end
1202

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
3
the node deleted is 200
1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
4
the node deleted is 1202
1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
3
the node deleted is 2
1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
5
contents of dq
100     190

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert      7:exit
enter the choice
7
Press any key to continue . . .
```

Lab Program 10:

```c
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<process.h>

struct node

 {

  int info;

  struct node *rlink;

  struct node *llink;

 };

typedef struct node *NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

 {

  printf("mem full\n");

  exit(0);

 }

 return x;

}

void freenode(NODE x)

{

free(x);

}

NODE insert(NODE root,int item)

{

NODE temp,cur,prev;

temp=getnode();
```

```c
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
          printf("  ");
  printf("%d\n",root->info);
          display(root->llink,i+1);
 }
}
void preorder(NODE root)
{
```

```c
if(root!=NULL)
 {
  printf("%d\n",root->info);
  preorder(root->llink);
  preorder(root->rlink);
 }
 }
void postorder(NODE root)
{
if(root!=NULL)
 {

  postorder(root->llink);
  postorder(root->rlink);
  printf("%d\n",root->info);
 }
 }
void inorder(NODE root)
{
if(root!=NULL)
 {

  inorder(root->llink);
  printf("%d\n",root->info);
  inorder(root->rlink);
 }
 }
void main()
{
int item,choice;
NODE root=NULL;
for(;;)
```

```c
{
printf("\n1.insert\t2.display\t3.preorder\t4.postorder\t5.inorder\t7.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item\n");
                scanf("%d",&item);
                root=insert(root,item);
                break;
  case 2:display(root,0);
                break;
  case 3:preorder(root);
                break;
  case 4:postorder(root);
                break;
  case 5:inorder(root);
                break;
  case 6:printf("enter the item\n");
                scanf("%d",&item);
                root=delete(root,item);
                break;
 default:exit(0);
                 break;
        }}}
```

```
1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
1
enter the item
50

1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
1
enter the item
23

1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
1
enter the item
45

1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
1
enter the item
6

1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
1
enter the item
89

1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
1
enter the item
13

1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
1
enter the item
177
```

```
1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
2
    177
  89
50
    45
  23
      13
    6
1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
3
50 23 6 13 45 89 177
1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
4
13 6 45 23 177 89 50
1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
5
6 13 23 45 50 89 177
1.insert        2.display       3.preorder      4.postorder     5.inorder       6.exit
enter the choice
9
wrong choice.THANK YOU..Press any key to continue . . .
```