B. M. S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Sep-2020 to Jan-2021

Data Structures Lab Record: -

Name: - Khushil M Sindhwad.

USN: - 1BM19CS072

Sec: - 3rd Sem CSE B

Batch: - B1

Lab-Program 1: -

Write a program to simulate the working of stack using an array with the following :
a) Push b) Pop c) Display
The program should print appropriate messages for stack overflow, stack underflow

Code: -

```c
#include <stdio.h>
#include <stdlib.h>
#define STACK_SIZE 5
int top=-1;
int s[10];
int item;
void push()
{
    if(top==STACK_SIZE-1)
    {
        printf("\nSTACK OVERFLOW\n");
        return;
    }
    top+=1;
    s[top]=item;
}
int pop()
{
    if(top==-1)
        return -1;
    return s[top--];
}
void display()
{
    int i;
    if(top==-1)
    {
        printf("\nSTACK UNDERFLOW\n");
        return;
    }
    printf("The contents fo the stack are:\n");
    for (i=0;i<=top;i++)
    {
        printf("%d\n",s[i]);
    }
}
void main()
{
    int item_deleted,choice;
    while(1)
    {
        printf("\n1: Push\n2: Pop\n3: Display\n4: EXIT\n");
        printf("Enter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
```

```
            printf("\nEnter the item to be inserted:");
            scanf("%d",&item);
            push();
            break;
            case 2:
            item_deleted=pop();
            if(item_deleted==-1)
            {
                printf("\nSTACK UNDERFLOW\n");
            }
            else
            {
                printf("\nThe item deleted is %d\n",item_deleted);
            }
            break;
            case 3:
            display();
            break;
            default:exit(0);


        }
    }

}
```

Output: -

```
1: Push
2: Pop
3: Display
4: EXIT
Enter your choice:1

Enter the item to be inserted:10

1: Push
2: Pop
3: Display
4: EXIT
Enter your choice:1

Enter the item to be inserted:20

1: Push
2: Pop
3: Display
4: EXIT
Enter your choice:1

Enter the item to be inserted:30

1: Push
2: Pop
3: Display
4: EXIT
Enter your choice:3
The contents fo the stack are:
10
20
30
```

```
1: Push
2: Pop
3: Display
4: EXIT
Enter your choice:2

The item deleted is 30

1: Push
2: Pop
3: Display
4: EXIT
Enter your choice:2

The item deleted is 20

1: Push
2: Pop
3: Display
4: EXIT
Enter your choice:3
The contents fo the stack are:
10

1: Push
2: Pop
3: Display
4: EXIT
Enter your choice:4
PS D:\C Programs>
```

Lab-Program 2: -

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

Code: -

```
#include<stdio.h>
#include<string.h>
int F(char symbol)
{
    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;
    }
}
void infix_postfix(char infix[],char postfix[])
{
    int top,i,j;
    char s[30],symbol;
    top=-1;
    s[++top]='#';
    j=0;
    for(i=0;i<strlen(infix);i++)
    {
        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';
}
void main()
{
    char infix[20];
    char postfix[20];
    printf("\nEnter the valid infix Expression:");
    scanf("%s",&infix);
    infix_postfix(infix,postfix);
    printf("\nThe postfix expression is: %s\n",postfix);
}
```

Output: -

```
Enter the valid infix Expression:((A+(B-C)*D)^E+F)

The postfix expression is: ABC-D*+E^F+
PS D:\C Programs> cd "d:\C Programs\" ; if ($?) { gcc

Enter the valid infix Expression:a^b*c-d+e/f/(g+h)

The postfix expression is: ab^c*d-ef/gh+/+
PS D:\C Programs> cd "d:\C Programs\" ; if ($?) { gcc

Enter the valid infix Expression:X^Y^Z-M+N+P/Q

The postfix expression is: XYZ^^M-N+PQ/+
```

Lab-Program 3: -

WAP to simulate the working of a queue of integers using an array. Provide the following operations
a) Insert b) Delete c) Display
The program should print appropriate messages for queue empty and queue overflow Conditions

Code: -

```
#include<stdio.h>
#include<process.h>
#define QUE_SIZE 3


int item,front=0,rear=-1,q[10];
void insertrear()
{
    if(rear==QUE_SIZE-1)
    {
        printf("\n-------------------------------\nQUEUE OVERFLOW\n-----------
--------------------\n");
        return;
    }
    rear+=1;
    q[rear]=item;

}

int deletefront()
{
    if(front>rear)
    {
        front=0;
        rear=-1;
        return -1;
    }
    return q[front++];
}

void displayQ()
{
    int i;
    if(front>rear)
    {
        printf("\n-----------------------------------\nQUEUE IS EMPTY\n-------
---------------------------\n");
        return;
    }
    printf("Contents of the queue:\n");
    for(i=front;i<=rear;i++)
    {
        printf("%d\n",q[i]);
    }
}
```

```c
void main()
{
    int choice;
    for(;;)
    {
        printf("\n1:Insert Rear\n2:Delete Front\n3:Display Queue\n4:EXIT\n");
        printf("Enter your choice:  ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("\nEnter the value to be inserted:  ");
                   scanf("%d",&item);
                   insertrear();
                   break;
            case 2:item=deletefront();
                   if(item==-1)
                       printf("\n-------------------------------------
\nQUEUE IS EMPTY\n-------------------------------------\n");
                   else
                       printf("Item Deleted= %d\n",item);
                   break;
            case 3:displayQ();
                   break;
            default:return;
        }
    }
}
```

Output: -

```
1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  1

Enter the value to be inserted:  12

1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  1

Enter the value to be inserted:  13

1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  1

Enter the value to be inserted:  14

1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  1

Enter the value to be inserted:  15

-----------------------------
QUEUE OVERFLOW
```

```
1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  3
Contents of the queue:
12
13
14

1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  2
Item Deleted= 12

1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  2
Item Deleted= 13

1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  2
Item Deleted= 14

1:Insert Rear
2:Delete Front
3:Display Queue
4:EXIT
Enter your choice:  2

-------------------------------------
QUEUE IS EMPTY
-------------------------------------
```

Lab-Program 4: -

WAP to simulate the working of a circular queue of integers using an array. Provide the following operations.
a) Insert b) Delete c) Display
The program should print appropriate messages for queue empty and queue overflow Conditions

Code: -

```
#include<stdio.h>
#include<process.h>
#define QUE_SIZE 3
int item,front=0,rear=-1,q[QUE_SIZE],count=0;
void insertrear()
{
    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\n");
        return;
    }
    f=front;
    printf("Contents of queue \n");
    for(i=1;i<=count;i++)
    {
        printf("%d\n",q[f]);
        f=(f+1)%QUE_SIZE;
    }
}
void main()
{
    int choice;
    for(;;)
    {
```

```
        printf("\n1:insertrear\n2:deletefront\n3:display\n4:exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);

        switch(choice)
        {
        case 1:printf("enter the item to be inserted\n");
            scanf("%d",&item);
            insertrear();
            break;
        case 2:item=deletefront();
            if(item==-1)
            printf("queue is empty\n");
            else
            printf("item deleted =%d\n",item);
            break;
        case 3:displayQ();
            break;
        default:exit(0);
        }
    }
}
```

Output: -

```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
12

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
13

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
14

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
15
queue overflow
```

```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
12
13
14

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =12

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
15

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
13
14
15
```

Lab-Program 5: -

WAP to Implement Singly Linked List with following operations
a) a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

Code: -

```c
#include <stdio.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 is 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)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty, Cannot Delete item\n");
```

```
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("Item Deleted at the front-end is: %d\n",first->info);
    free(first);
    return temp;
}

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;
}
NODE delete_rear(NODE first)
{
    NODE cur,prev;
    if(first==NULL)
    {
        printf("The List is Empty, Cannot Delete Item\n");
        return first;
    }
    if(first->link==NULL)
    {
        printf("Item Deleted is: %d",first->info);
        free(first);
        return NULL;
    }
    prev=NULL;
    cur=first;
    while(cur->link!=NULL)
    {
        prev=cur;
        cur=cur->link;
    }
    printf("Item Deleted at the rear-end is : %d",cur->info);
    free(cur);
    prev->link=NULL;
    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 Position\n");
        return first;
    }
    if(pos==1)
    {
        temp->link=first;
        return temp;
    }
    count=1;
    prev=NULL;
    cur=first;
    while(cur!=NULL && count!=pos)
    {
        prev=cur;
        cur=cur->link;
        count++;
    }
    if(count==pos)
    {
        prev->link=temp;
        temp->link=cur;
        return first;
    }
    printf("IP\n");
    return first;
}

void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is EMPTY , Cannot Display Items\n");
        return;
    }
    printf("\n*****************************************************************\
n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
    printf("\n*****************************************************************\
n");

}

void main()
{
    int item,choice,pos;
    NODE first=NULL;
```

```
    for(;;)
    {
        printf("\n1:Insert_front\n2:Delete_front\n3:Insert_rear\n4:Delete_rear
\n5:insert_pos\n6:display_list\n7:Exit\n");
        printf("Enter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the item at front-end: ");
                scanf("%d",&item);
                first=insert_front(first,item);
                break;
            case 2:first=delete_front(first);
                break;
            case 3:printf("Enter the item at rear-end: ");
                scanf("%d",&item);
                first=insert_rear(first,item);
                break;
            case 4:first=delete_rear(first);
                break;
            case 5:printf("Enter the position: ");
                scanf("%d",&pos);
                first=insert_pos(item,pos,first);
                break;
            case 6:display(first);
                break;
            default:exit(0);
                break;
        }
    }
}
```

Output: -

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:insert_pos
6:display_list
7:Exit
Enter the choice: 1
Enter the item at front-end: 10

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:insert_pos
6:display_list
7:Exit
Enter the choice: 3
Enter the item at rear-end: 20

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:insert_pos
6:display_list
7:Exit
Enter the choice: 1
Enter the item at front-end: 30

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:insert_pos
6:display_list
7:Exit
Enter the choice: 6

************************************************************
30
10
20

************************************************************
```

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:insert_pos
6:display_list
7:Exit
Enter the choice: 2
Item Deleted at the front-end is: 30

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:insert_pos
6:display_list
7:Exit
Enter the choice: 4
Item Deleted at the rear-end is : 20
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:insert_pos
6:display_list
7:Exit
Enter the choice: 6

************************************************************
10

************************************************************

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:insert_pos
6:display_list
7:Exit
Enter the choice: 7
PS D:\DS 3rd Sem Notes\DS Lab\Week 8>
```

Lab-Program 6: -

WAP to Implement Singly Linked List with following operations
a) a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

Code: -

```
#include <stdio.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 is 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)
{
    NODE temp;
    if(first==NULL)
```

```
    {
        printf("List is empty, Cannot Delete item\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("Item Deleted at the front-end is: %d\n",first->info);
    free(first);
    return temp;
}

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;
}
NODE delete_rear(NODE first)
{
    NODE cur,prev;
    if(first==NULL)
    {
        printf("The List is Empty, Cannot Delete Item\n");
        return first;
    }
    if(first->link==NULL)
    {
        printf("Item Deleted is: %d",first->info);
        free(first);
        return NULL;
    }
    prev=NULL;
    cur=first;
    while(cur->link!=NULL)
    {
        prev=cur;
        cur=cur->link;
    }
    printf("Item Deleted at the rear-end is : %d",cur->info);
    free(cur);
    prev->link=NULL;
    return first;
}
```

```
NODE delete_pos(int pos,NODE first)
{
    NODE prev,cur;
    int count;
    if(first==NULL || pos<=0)
    {
        printf("Invalid position\n");
        return NULL;
    }
    if(pos==1)
    {
        cur=first;
        first=first->link;
        freenode(cur);
        return first;
    }
    prev=NULL;
    cur=first;
    count=1;
    while(cur!=NULL)
    {
        if(count==pos)
        {
            break;
        }
        prev=cur;
        cur=cur->link;
        count++;
    }
    if(count!=pos)
    {
        printf("Invalid Position\n");
        return first;
    }
    prev->link=cur->link;
    freenode(cur);
    return first;
}


void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is EMPTY , Cannot Display Items\n");
        return;
    }
    printf("\n*****************************************************************\
n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
    printf("\n*****************************************************************\
n");
```

```
}

void main()
{
    int item,choice,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("\n1:Insert_front\n2:Delete_front\n3:Insert_rear\n4:Delete_rear
\n5:delete_pos\n6:display_list\n7:Exit\n");
        printf("Enter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the item at front-end: ");
                scanf("%d",&item);
                first=insert_front(first,item);
                break;
            case 2:first=delete_front(first);
                break;
            case 3:printf("Enter the item at rear-end: ");
                scanf("%d",&item);
                first=insert_rear(first,item);
                break;
            case 4:first=delete_rear(first);
                break;
            case 5:printf("Enter the position: ");
                scanf("%d",&pos);
                first=delete_pos(pos,first);
                break;
            case 6:display(first);
                break;
            default:exit(0);
                break;
        }
    }
}
```

Output: -

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:delete_pos
6:display_list
7:Exit
Enter the choice: 1
Enter the item at front-end: 12

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:delete_pos
6:display_list
7:Exit
Enter the choice: 1
Enter the item at front-end: 13

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:delete_pos
6:display_list
7:Exit
Enter the choice: 1
Enter the item at front-end: 14

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:delete_pos
6:display_list
7:Exit
Enter the choice: 1
Enter the item at front-end: 15
```

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:delete_pos
6:display_list
7:Exit
Enter the choice: 6

************************************************************
15
14
13
12

************************************************************

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:delete_pos
6:display_list
7:Exit
Enter the choice: 2
Item Deleted at the front-end is: 15

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:delete_pos
6:display_list
7:Exit
Enter the choice: 5
Enter the position: 3

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:delete_pos
6:display_list
7:Exit
Enter the choice: 6

************************************************************
14
13

************************************************************
```

Lab-Program 7: -

WAP Implement Single Link List with following operations
a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists

Code: -

```c
#include<stdio.h>
#include<stdlib.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);
}
void display (NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("Linked List is empty ,Cannot Display items");
        return;
    }
    printf("The contents of the linked list are: \n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }

}

NODE order_list(int item,NODE first)
{
    NODE temp,prev,cur;
    temp=getnode();
    temp->info=item;
```

```
    temp->link=NULL;
    if(first==NULL) return temp;
    if(item<first->info)
    {
        temp->link=first;
        return temp;
    }
    prev=NULL;
    cur=first;
    while(cur!=NULL && item>cur->info)
    {
        prev=cur;
        cur=cur->link;
    }
    prev->link=temp;
    temp->link=cur;
    return first;
}
NODE concat(NODE first,NODE second)
{
    NODE cur;
    if(first==NULL)
        return second;
    if(second==NULL)
        return first;
    cur=first;
    while(cur->link!=NULL)
        cur=cur->link;
    cur->link=second;
    return first;
}

NODE reverse(NODE first)
{
    NODE cur,temp;
    cur=NULL;
    while(first!=NULL)
    {
        temp=first;
        first=first->link;
        temp->link=cur;
        cur=temp;
    }
    return cur;
}

int main()
{
    NODE first=NULL;
    NODE second=NULL;
    int item, choice,llno;
    for(;;)
    {
        printf("1:insert in order list\n2:Concatenate the two linked lists\n3:
reverse the linked list\n4:Display\n5:EXIT\n");
        printf("Enter your choice: ");
```

```
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the LL number (1 or 2) :");
                    scanf("%d",&llno);
                    printf("Enter the item at the rear end: ");
                    scanf("%d",&item);
                    if(llno==1)
                        first=order_list(item,first);
                    else
                        second=order_list(item,second);
                    break;

            case 2: first=concat(first,second);
                    second=NULL;
                    break;


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

            case 4: printf("The contents of the first LL:\n");
                    display(first);
                    printf("\n*******************************\n");
                    printf("\nThe contents of the second LL:\n");
                    display(second);
                    break;

            case 5: exit(0);
                    break;

            default: printf("Enter a valid option\n");
        }
        printf("\n");
    }
    return 0;
}
```

Output: -

```
1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 1
Enter the LL number (1 or 2) :1
Enter the item at the rear end: 12

1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 1
Enter the LL number (1 or 2) :1
Enter the item at the rear end: 15

1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 1
Enter the LL number (1 or 2) :1
Enter the item at the rear end: 14

1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 1
Enter the LL number (1 or 2) :2
Enter the item at the rear end: 16

1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 1
Enter the LL number (1 or 2) :2
Enter the item at the rear end: 19
```

```
1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 1
Enter the LL number (1 or 2) :2
Enter the item at the rear end: 18

1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 4
The contents of the first LL:
The contents of the linked list are:
12
14
15

*******************************

The contents of the second LL:
The contents of the linked list are:
16
18
19

1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 2

1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 4
```

```
4:Display
5:EXIT
Enter your choice: 4
The contents of the first LL:
The contents of the linked list are:
12
14
15
16
18
19

*********************************

The contents of the second LL:
Linked List is empty ,Cannot Display items
1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 3

1:insert in order list
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 4
The contents of the first LL:
The contents of the linked list are:
19
18
16
15
14
12

*********************************
```

Lab-Program 8: -

WAP to implement Stack & Queues using Linked Representation

Code: -

Stacks: -

```c
#include <stdio.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 is 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)
{
    NODE temp;
    if(first==NULL)
    {
```

```
        printf("List is empty, Cannot Delete item\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("Item Deleted at the front-end is: %d\n",first->info);
    free(first);
    return temp;
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is EMPTY , Cannot Display Items\n");
        return;
    }
    printf("\n*************************************************************\
n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
    printf("\n*************************************************************\
n");

}

void main()
{
    int item,choice,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("\n1:PUSH\n2:POP\n3:insert_pos\n4:display_list\n5:Exit\n");
        printf("Enter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the item : ");
                scanf("%d",&item);
                first=insert_front(first,item);
                break;
            case 2:first=delete_front(first);
                break;
            case 3:printf("Enter the position: ");
                scanf("%d",&pos);
                first=insert_pos(item,pos,first);
                break;
            case 4:display(first);
                break;
            default:exit(0);
                break;
        }
    }
}
```

Output for Stacks: -

```
 1:PUSH
 2:POP
 3:insert_pos
 4:display_list
 5:Exit
 Enter the choice: 1
 Enter the item at front-end: 10

 1:PUSH
 2:POP
 3:insert_pos
 4:display_list
 5:Exit
 Enter the choice: 1
 Enter the item at front-end: 20

 1:PUSH
 2:POP
 3:insert_pos
 4:display_list
 5:Exit
 Enter the choice: 1
 Enter the item at front-end: 30

 1:PUSH
 2:POP
 3:insert_pos
 4:display_list
 5:Exit
 Enter the choice: 4

 ********************************************************************
 30
 20
 10

 ********************************************************************
1:PUSH
2:POP
3:insert_pos
4:display_list
5:Exit
Enter the choice: 2
Item Deleted at the front-end is: 30

1:PUSH
2:POP
3:insert_pos
4:display_list
5:Exit
Enter the choice: 2
Item Deleted at the front-end is: 20

1:PUSH
2:POP
3:insert_pos
4:display_list
5:Exit
Enter the choice: 2
Item Deleted at the front-end is: 10

1:PUSH
2:POP
3:insert_pos
4:display_list
5:Exit
Enter the choice: 2
List is empty, Cannot Delete item
```

Code for Queues: -

```c
#include <stdio.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 is full\n");
        exit(0);
    }
    return x;
}

void freenode(NODE x)
{
    free(x);
}
NODE delete_front(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty, Cannot Delete item\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("Item Deleted at the front-end is: %d\n",first->info);
    free(first);
    return temp;
}

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)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is EMPTY , Cannot Display Items\n");
        return;
    }
    printf("\n*************************************************************\
n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
    printf("\n*************************************************************\
n");

}

void main()
{
    int item,choice,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("\n1:Insert_rear\n2:Delete_front\n3:display_Queue\n4:Exit\n");
        printf("Enter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the item at rear-end: ");
                scanf("%d",&item);
                first=insert_rear(first,item);
                break;
            case 2:first=delete_front(first);
                break;
            case 3:display(first);
                break;
            default:exit(0);
                break;
        }
    }
}
```

Output for Queues: -

```
1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 1
Enter the item at rear-end: 12

1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 1
Enter the item at rear-end: 13

1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 1
Enter the item at rear-end: 14

1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 3

*****************************************************************
12
13
14

*****************************************************************

1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 2
Item Deleted at the front-end is: 12

1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 2
Item Deleted at the front-end is: 13

1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 2
Item Deleted at the front-end is: 14

1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 2
List is empty, Cannot Delete item

1:Insert_rear
2:Delete_front
3:display_Queue
4:Exit
Enter the choice: 4
PS D:\DS 3rd Sem Notes\DS Lab\Week 8>
```

Lab-Program 9: -

WAP Implement doubly link list with primitive operations
a) a) Create a doubly linked list. b) Insert a new node to the left of the node.
b) c) Delete the node based on a specific value. c) Display the contents of the list

Code: -

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

struct node
{
    int info;
    struct node *llink;
    struct node *rlink;
};
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 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_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)
{
    NODE cur,next;
    if(head->rlink==head)
    {
        printf("The DLL is 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("The DLL is 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;
}
NODE insert_leftpos(int item,NODE head)
{
    NODE temp,cur,prev;
    int item2;
    if(head->rlink==head)
    {
        printf("List empty\n");
        return head;
    }
    cur=head->rlink;
    while(cur!=head)
    {
        if(item==cur->info)break;
        cur=cur->rlink;
    }
    if(cur==head)
    {
        printf("key not found\n");
        return head;
```

```
    }
    prev=cur->llink;
    temp=getnode();
    printf("Enter towards left of %d : ",item);
    scanf("%d",&item2);
    temp->info=item2;
    prev->rlink=temp;
    temp->llink=prev;
    cur->llink=temp;
    temp->rlink=cur;
    return head;
}
NODE insert_rightpos(int item,NODE head)
{
    NODE temp,cur,next;
    int item2;
    if(head->rlink==head)
    {
        printf("List empty\n");
        return head;
    }
    cur=head->rlink;
    while(cur!=head)
    {
        if(item==cur->info)break;
        cur=cur->rlink;
    }
    if(cur==head)
    {
        printf("key not found\n");
        return head;
    }
    next=cur->rlink;
    temp=getnode();
    printf("Enter towards right of %d : ",item);
    scanf("%d",&item2);
    temp->info=item2;
    next->llink=temp;
    temp->rlink=next;
    cur->rlink=temp;
    temp->llink=cur;
    return head;
}

NODE delete_all_key(int item, NODE head)
{
    NODE prev,cur,next;
    int count;
    if(head->rlink==head)
    {
        printf("List empty\n");
        return head;
    }
    count=0;
    cur=head->rlink;
    while(cur!=head)
```

```
    {
        if(item!=cur->info)
            cur=cur->rlink;
        else
        {
            count++;
            prev=cur->llink;
            next=cur->rlink;
            prev->rlink=next;
            next->llink=prev;
            freenode(cur);
            cur=next;
        }
    }
    if(count==0)
        printf("Key not found\n");
    else
        printf("Key found at %d positions !! and are deleted",count);
    return head;

}

void searching(int key,NODE head)
{
    NODE temp,cur;
    if(head->rlink==head)
    {
        printf("list empty\n");
        return;
    }
    cur=head->rlink;
    while(cur!=head)
    {
        if(cur->info==key)
        {
            printf("Search Successful\n");
            return;
        }
        cur=cur->rlink;
    }
    printf("Search is not successfull\n");
    return;
}
NODE ddelete_duplicates(int item,NODE head)
{
    NODE prev,cur,next;
    int count=0;
    if (head->rlink==head)
    {
        printf("List is empty\n");
        return head;
    }
    cur=head->rlink;
    while (cur!=head)
    {
        if (cur->info!=item)
```

```
        {
            cur=cur->rlink;
        }
        else
        {
            count++;
            if (count==1)
            {
                cur=cur->rlink;
                continue;
            }
            else
            {
                prev=cur->llink;
                next=cur->rlink;
                prev->rlink=next;
                next->llink=prev;
                free(cur);
                cur=next;
            }
        }
    }
    if (count==0)
    {
        printf("No such item found in the list\n");
    }
    else
    {
        printf("Removed all the duplicate elements of the given item successfu
lly\n");
    }
    return head;
}
void display(NODE head)
{
    NODE temp;
    if(head->rlink==head)
    {
        printf("The DLL is empty");
        return;
    }
    printf("the contents of the DLL\n");
    temp=head->rlink;
    while(temp!=head)
    {
        printf("%d\n",temp->info);
        temp=temp->rlink;
    }
    printf("\n");
}

void main()
{
    NODE head,last;
    int item,choice;
    head=getnode();
```

```
    head->rlink=head;
    head->llink=head;
    for(;;)
    {
        printf("\n1:Insert front\n2:Insert rear\n3:Delete front\n4:Delete rear
\n5:Insert_key_Left\n6:Insert_key_Right\n7:Delete all keys\n8:Search item\n9:D
elete Only duplicates\n10:Display\n11:Exit\n");
        printf("Enter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the item at front end : ");
                    scanf("%d",&item);
                    last=dinsert_front(item,head);
                    break;
            case 2: printf("Enter the item at rear end : ");
                    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:printf("Enter the key item: ");
                    scanf("%d",&item);
                    head=insert_leftpos(item,head);
                    break;
            case 6:printf("Enter the key item: ");
                    scanf("%d",&item);
                    head=insert_rightpos(item,head);
                    break;
            case 7:printf("Enter the key item: ");
                    scanf("%d",&item);
                    head=delete_all_key(item,head);
                    break;
            case 8:printf("Enter the key item: ");
                    scanf("%d",&item);
                    searching(item,head);
                    break;
            case 9:printf("Enter the key item: ");
                    scanf("%d",&item);
                    head=ddelete_duplicates(item,head);
                    break;
            case 10: display(head);
                    break;
            default: return;
        }
    }
}
```

Output: -

```
1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 1
Enter the item at front end : 12

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 2
Enter the item at rear end : 13

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 2
Enter the item at rear end : 14

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
```

```
11:Exit
Enter the choice: 2
Enter the item at rear end : 15

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 5
Enter the key item: 14
Enter towards left of 14 : 12

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 10
the contents of the DLL
12
13
12
14
15


1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 6
Enter the key item: 15
```

```
Enter towards right of 15 : 12

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 10
the contents of the DLL
12
13
12
14
15
12


1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 7
Enter the key item: 12
Key found at 3 positions !! and are deleted
```

```
1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 10
the contents of the DLL
13
14
15


1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 8
Enter the key item: 14
Search Successful

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert_key_Left
6:Insert_key_Right
7:Delete all keys
8:Search item
9:Delete Only duplicates
10:Display
11:Exit
Enter the choice: 11
PS D:\DS 3rd Sem Notes\DS Lab\Week 10>
```

Lab-Program 10: -

Write a program
a) To construct a binary Search tree.
b) To traverse the tree using all the methods i.e., in-order, preorder and post order
c) To display the elements in the tree.

Code: -

```
#include<stdio.h>
#include<stdlib.h>
#include<string.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();
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);
 }
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
 printf("not found\n");
 return root;
}
if(cur->llink==NULL)
 q=cur->rlink;
else if(cur->rlink==NULL)
 q=cur->llink;
else
 {
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

void preorder(NODE root)
```

```
{
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(;;)
    {
        printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.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;
                }
            }
        }
```

Output: -

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
12

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
15

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
14

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
16
```

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
9

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    16
  15
    14
12
  10
    9

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
```

```
enter the choice
3
12
10
9
15
14
16

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
9
10
14
16
15
12

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
9
10
12
14
15
16

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
```

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
14

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    16
  15
12
  10
    9

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
7
```