

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



## **LAB REPORT on**

## **DATA STRUCTURES**

*Submitted by*

**SONY MAHATO (1BM21CS217)**

*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**  
**Oct 2022-Feb 2023**

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



**CERTIFICATE**

This is to certify that the Lab work entitled “**DATA STRUCTURES**” carried out by **SONY MAHATO (1BM21CS217)**, who is 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 Visvesvaraya Technological University, Belgaum during the year 2022-23. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab - (**22CS3PCDST**) work prescribed for the said degree.

Name of the Lab-Incharge  
Designation  
Department of CSE  
BMSCE, Bengaluru

**Dr. Jyothi S Nayak**  
Professor and Head  
Department of CSE  
BMSCE, Bengaluru

## Index Sheet

Sl. No.	Experiment Title	Page No.
1	Implementation of Stack without using pointers	4
2	Conversion of Infix Expression to Postfix Expression	8
3	Implementation of Linear Queue using array	12
4	Implementation of Circular Queue using array	17
5	Creation of Linked List and displaying the contents	23
6	Implementation of Linked List with creation, insertion and displaying the contents	25
7	Implementation of Singly Linked List with creation, deletion and displaying the contents	32
8	Implementation of Singly Linked List with sorting, reversing and concatenation of two linked lists	38
9	Implementation of Stacks and Queues using Linked Representation	45
10	Implementation of Doubly Linked List with primitive operations	52

## Course Outcome

C01	Apply the concept of linear and nonlinear data structures.
C02	Analyse data structure operations for a given problem.
C03	Design and develop solutions using the operations of linear and nonlinear data structure for a given specification.
C04	Conduct practical experiments for demonstrating the operations of different data structures.

## **LAB PROGRAM 1: Implementation of Stack without using pointers**

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.

Program:

```
#include<stdio.h>
#include<conio.h>
#define SIZE 3
int STACK[SIZE],TOP=-1,ITEM;
void push();
void pop();
void display();

int main()
{
    int choice;
    while(1)
    {
        if("\n\n 1:Push\n 2:Pop \n 3: Display\n 4:Exit\n");
        printf("Enter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:push();
            break;
            case 2:pop();
            break;
```

```

        case 3:display();
        break;
        case 4:exit();
        break;
        default:printf("Wrong Choice");
    }
}
getch();
return(0);
}

void push()
{
    if(TOP==SIZE-1)
    {
        printf("Stack Overflow");
        return;
    }
    else
    {
        printf("Enter an element\n");
        scanf("%d",&ITEM);
        printf("Entered element is %d\n\n",ITEM);
        TOP=TOP+1;
        STACK[TOP]=ITEM;
    }
}

void pop()
{
    int del;
    if(TOP==-1)
    {
        printf("Stack Underflow");
    }
}

```

```

        return;
    }
    else
    {
        del=STACK[TOP];
        printf("Popped element is %d\n",del);
        TOP=TOP-1;
    }
}

void display()
{
    int i;
    if(TOP== -1)
    {
        printf("Stack is Empty\n");
        return;
    }
    else
    {
        for(i=TOP;i>=0;i--)
            printf("%d\n",STACK[i]);
    }
}

```

Output:

```
C:\Users\BMSCE\Desktop\1bm21cs213\stack.exe
Enter your choice:1
Enter an element
10
Entered element is 10

Enter your choice:1
Enter an element
20
Entered element is 20

Enter your choice:1
Enter an element
30
Entered element is 30

Enter your choice:3
30
20
10
Enter your choice:2
Popped element is 30
Enter your choice:3
20
10
Enter your choice:2
Popped element is 20
Enter your choice:3
10
Enter your choice:2
Popped element is 10
Enter your choice:3
Stack is Empty
Enter your choice:4

Process returned 0 (0x0)   execution time : 26.657 s
Press any key to continue.
```

## **LAB PROGRAM 2: Conversion of Infix Expression to Postfix Expression**

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)

Program:

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

int index=0, pos=0, top=-1, length;
char symbol, temp, infix[20], postfix[20], stack[20];

void infixtopostfix();
void push(char symbol);
char pop();
int pred(char symb);

void main()
{
    printf("Enter Infix Expression:\n");
    scanf("%s",infix);

    infixtopostfix();

    printf("\nInfix expression:\n%s",infix);
    printf("\nPostfix expression:\n%s",postfix);

    getch();
}
```



```

void infixtopostfix()
{
    length=strlen(infix);
    push('#');
    while(index<length)
    {
        symbol=infix[index];
        switch(symbol)
        {
            case '(':push(symbol);
                break;
            case ')':temp=pop();
                while(temp!='(')
                {
                    postfix[pos]=temp;
                    pos++;
                    temp=pop();
                }
                break;
            case '+':
            case '-':
            case '*':
            case '/':
            case '^': while(pred(stack[top])>=pred(symbol))
                {
                    temp=pop();
                    postfix[pos++]=temp;
                }
                push(symbol);
                break;
            default: postfix[pos++]=symbol;
        }
    }
}

```

```

        index++;
    }
    while(top>0)
    {
        temp=pop();
        postfix[pos++]=temp;
    }
}

```

```

void push(char symbol)
{
    top=top+1;
    stack[top]=symbol;
}

```

```

char pop()
{
    char symb;
    symb=stack[top];
    top=top-1;
    return(symb);
}

```

```

int pred(char symbol)
{
    int p;
    switch(symbol)
    {
        case '^':p=3;
            break;
        case '*':
        case '/':p=2;
    }
}

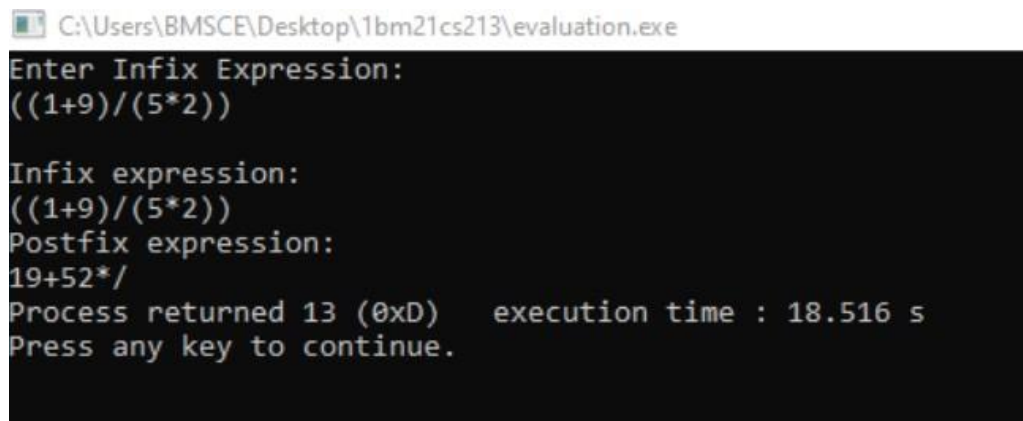
```

```

        break;
    case '+':
    case '-':p=1;
        break;
    case '(':p=0;
        break;
    case '#':p=-1;
        break;
}
return(p);
}

```

Output:



The screenshot shows a Windows command prompt window with the title bar "C:\Users\BMSCE\Desktop\1bm21cs213\evaluation.exe". The text inside the window is as follows:

```

Enter Infix Expression:
((1+9)/(5*2))

Infix expression:
((1+9)/(5*2))
Postfix expression:
19+52*/
Process returned 13 (0xD)    execution time : 18.516 s
Press any key to continue.

```

### **LAB PROGRAM 3: Implementation of Linear Queue using array**

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.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#define size 3
int queue[size];
int front=-1; int rear=-1;
void insert();
void delete();
void display();

void main(){
    int choice;
    while(1){
        printf("\n Enter choice \n 1.Insert \n 2.Delete \n 3.Display \n 4.Exit");
        scanf("%d",&choice);
        switch(choice){
            case 1:insert();
                break;
            case 2:delete();
                break;
            case 3:display();
                break;
            case 4:exit(0);
```

```

        default:printf("Invalid choice:");
    }
}
return 0;
}

```

```

void insert(){
    int val;
    if(rear==size-1||rear==front-1){
        printf("Stack overflow");
    }
    else{
        printf("Enter a value:");
        scanf("%d",&val);
        if(front==-1&&rear==-1){
            front=rear=0;
        }
        else{
            rear+=1;
        }
    }
    queue[rear]=val;
}

```

```

void delete(){
    int val;
    if(front==-1&&rear==-1){
        printf("Queue is empty");
    }
    val=queue[front];
    if(front==rear){
        front=rear=-1;
    }
}

```

```

    }
    else{
        front+=1;
    }
}

void display(){
    if(front==-1&&rear==-1){
        printf("\nQueue is empty");
    }
    for(int i=front;i<=rear;i++){
        printf("%d\t",queue[i]);
    }
}

```

Output:

```
Enter choice
1.Insert
2.Delete
3.Display
4.Exit
3
Queue is empty
Enter choice
1.Insert
2.Delete
3.Display
4.Exit
1
Enter a value:10

Enter choice
1.Insert
2.Delete
3.Display
4.Exit
1
Enter a value:20

Enter choice
1.Insert
2.Delete
3.Display
4.Exit
1
Enter a value:30

Enter choice
1.Insert
2.Delete
3.Display
4.Exit
3
10      20      30
```

```
Enter choice
1.Insert
2.Delete
3.Display
4.Exit
2

Enter choice
1.Insert
2.Delete
3.Display
4.Exit
3
20      30
Enter choice
1.Insert
2.Delete
3.Display
4.Exit
2

Enter choice
1.Insert
2.Delete
3.Display
4.Exit
3
30
Enter choice
1.Insert
2.Delete
3.Display
4.Exit
2

Enter choice
1.Insert
2.Delete
3.Display
4.Exit3
Queue is empty0
Enter choice
1.Insert
2.Delete
3.Display
4.Exit
```



## **LAB PROGRAM 4: Implementation of Circular Queue using array**

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.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#define max 6
int queue[max];
int front=-1;
int rear=-1;

void enqueue(int element)
{
    if(front== -1 && rear == -1)
    {
        front=0;
        rear=0;
        queue[rear]=element;
    }
    else if((rear+1)%max==front)
    {
        printf("queue is overflow");
    }
    else{
        rear=(rear+1)%max;
        queue[rear]=element;
    }
}
```

```

    }
}

int dequeue()
{
    if((front== -1) && (rear== -1))
    {
        printf("\n queue is underflow");
    }
    else if(front==rear)
    {
        printf("\n the dequeued element is %d", queue[front]);
        front=-1;
        rear=-1;
    }
    else{
        printf("\n the dequeued element is %d", queue[front]);
        front=(front+1)%max;
    }
}

void display()
{
    int i=front;
    if(front== -1 && rear== -1)
    {
        printf("\n queue is empty");
    }
    else
    {
        printf("\n elements in a queue are:");
        while(i<=rear)
        {
            printf("%d\n", queue[i]);

```

```

        i=(i+1)%max;
    }
}

int main()
{
    int choice=1,x;
    while(1)
    {
        printf("\n 1. insert an element\n");
        printf("\n 2. delete an element\n");
        printf("\n 3. display all elements\n");
        printf("\n 4. exit \n");
        printf("\n enter your choice");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1 : printf("\n enter element to be inserted\n");
                     scanf("%d",&x);
                     enqueue(x);
                     break;
            case 2 : dequeue();
                     break;
            case 3: display();
                     break;
            case 4: exit(0);
                     break;
            default : printf("enter a valid choice");
        }
    }
    return(0);
}

```

Output:

```
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice1
enter element to be inserted
10

1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice1
enter element to be inserted
20

1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice1
enter element to be inserted
30

1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice3
elements in a queue are:10
20
30
```

```
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice2
the dequeued element is 10
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice3
elements in a queue are:20
30
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice2
the dequeued element is 20
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice3
elements in a queue are:30
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice2
the dequeued element is 30
```

```
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice3
queue is empty
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice2
queue is underflow
1. insert an element
2. delete an element
3. display all elements
4. exit
enter your choice4
Process returned 0 (0x0)   execution time : 36.126 s
Press any key to continue.
```

## **LAB PROGRAM 5: Creation of Linked List and displaying the contents**

WAP to create Linked List and display the contents of Linked List

Program:

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

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

int main()
{
struct node *head= malloc(sizeof(struct node));
head->data = 45;
head->link= NULL;

struct node *current = malloc(sizeof(struct node));
current->data= 55;
current->link=NULL;
head->link=current;

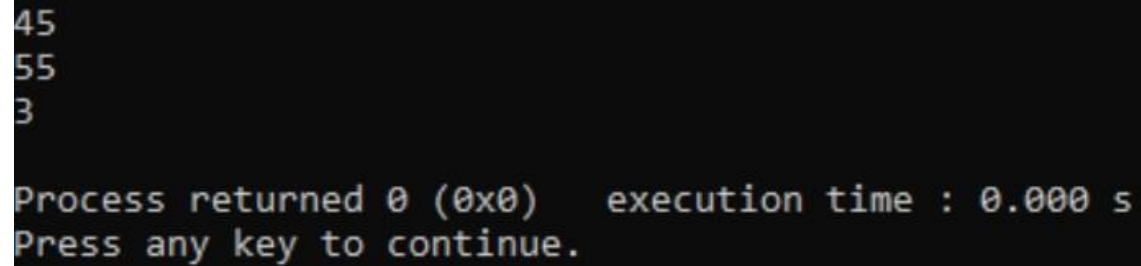
current = malloc(sizeof(struct node));
current->data= 3;
current->link=NULL;
head->link->link=current;

print_data(head);
}
```

```
void print_data(struct node *head)
{
    if(head==NULL)
        printf("linked list is empty");
    struct node*ptr=NULL;
    ptr = head;

    while(ptr!=NULL)
    {
        printf("%d\n", ptr->data);
        ptr= ptr->link;
    }
}
```

Output:



```
45
55
3

Process returned 0 (0x0)   execution time : 0.000 s
Press any key to continue.
```



## **LAB PROGRAM 6: Implementation of Linked List with creation, insertion and displaying the contents**

WAP to Implement Singly Linked List with following operations

- 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.

Program:

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

void create();
void display();
void insert_head();
void insert_last();
void insert_val();

struct Node
{
    int data;
    struct Node *link;
};
typedef struct Node node;
node *start=NULL;

int main()
{
    int ch;
    while(1)
    {
```

```

        printf("\n1.Create\n2.Display \n3.Insert Head \n4.Insert
Last\n5.Insert val\n6.Exit");

        printf("\nEnter your choice:\n");

        scanf("%d",&ch);

        switch(ch)
        {
        case 1:
            create();
            break;
        case 2:
            display();
            break;
        case 3:
            insert_head();
            break;
        case 4:
            insert_last();
            break;
        case 5: insert_val();
            break;
        case 6:
            exit(1);

        default:
            printf("Invalid choice\n");
        }
    }

    return 0;
}

void create()
{
    int c;

```

```

node *neww,*curr;

start=(node *) malloc(sizeof(node));

curr=start;

printf("Enter element\n");

scanf("%d",&start->data);

while(1)
{
    printf("Do you want to add another element(1/0)\n");
    scanf("%d",&c);
    if(c==1)
    {
        neww=(node *) malloc(sizeof(node));
        printf("Enter element\n");
        scanf("%d",&neww->data);
        curr->link = neww;
        curr=neww;
    }
    else
    {
        curr->link=NULL;
        break;
    }
}

void display()
{
    node *temp;

    if(start==NULL)
    {
        printf("Linked list is empty\n");
        return;
    }
}

```

```

temp=start;
while(temp!=NULL)
{
    printf("%d\t",temp->data);
    temp = temp->link;
}
}

void insert_head(){
    node *temp,*mew;
    mew = (node *) malloc(sizeof(node));
    temp = start;
    printf("enter element value");
    scanf("%d",&mew->data);
    mew->link = start;
    start = mew;
}

void insert_last(){
    node *neww,*temp;
    neww = (node *) malloc(sizeof(node));
    temp = start;
    printf("enter element value");
    scanf("%d",&neww->data);
    while(temp->link!=NULL)
    {
        temp = temp->link;
    }
    temp->link = neww;
    neww->link = NULL;
}

void insert_val(){
    int pos;
    node *neww, *temp;

```

```

neww =(node*)malloc(sizeof(node));
printf("\nEnter element: ");
scanf("%d",&neww->data);
printf("Enter position\n");
scanf("%d",&pos);
if(pos==1)
{
    neww->link=start;
    start=neww;
    return;
}
int i=1;
temp=start;
while(i<(pos-1) && temp!=NULL)
{
    temp=temp->link;
    i++;
}
if(i==(pos-1))
{
    neww->link=temp->link;
    temp->link=neww;
    return;
}
if(temp==NULL)
{
    printf("Invalid position\n");
}
}

```

Output:

```
1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
1
Enter element
10
Do you want to add another element(1/0)
1
Enter element
20
Do you want to add another element(1/0)
1
Enter element
30
Do you want to add another element(1/0)
0
```

```
1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
2
10      20      30
1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
3
enter element value5
```

```
1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
2
5        10      20      30
1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
4
enter element value40
```

```

1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
2
5      10      20      30      40
1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
5
Enter element: 50
Enter position
3
1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
2
5      10      50      20      30      40
1.Create
2.Display
3.Insert Head
4.Insert Last
5.Insert val
6.Exit
Enter your choice:
6
Process returned 1 (0x1)  execution time : 50.672 s
Press any key to continue.

```

## **LAB PROGRAM 7: Implementation of Singly Linked List with creation, deletion and displaying the contents**

WAP to Implement Singly Linked List with following operations

- 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.

Program:

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

void create();
void display();
void delete_head();
void delete_last();
void delete_val();

struct Node
{
    int data;
    struct Node *link;
};
typedef struct Node node;
node *start=NULL;

int main()
{
    int ch;
    while(1)
    {
```



```

        printf("\n1.Create\n2.Display\n3.Delete Head\n4.Delete Last\n5.Delete
val\n6.Exit");

        printf("\nEnter your choice:\n");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1:
            create();
            break;
        case 2:
            display();
            break;
        case 3: delete_head();
            break;
        case 4:delete_last();
            break;
        case 5:delete_val();
            break;
        case 6:
            exit(1);

        default:
            printf("Invalid choice\n");
        }
    }
    return 0;
}

void create()
{
    int c;
    node *neww,*curr;
    start=(node *) malloc(sizeof(node));

```

```

curr=start;
printf("Enter element\n");
scanf("%d",&start->data);
while(1)
{
    printf("Do you want to add another element(1/0)\n");
    scanf("%d",&c);
    if(c==1)
    {
        neww=(node *) malloc(sizeof(node));
        printf("Enter element\n");
        scanf("%d",&neww->data);
        curr->link = neww;
        curr=neww;
    }
    else
    {
        curr->link=NULL;
        break;
    }
}
}

```

```

void display()
{
    node *temp;
    if(start==NULL)
    {
        printf("Linked list is empty\n");
        return;
    }
    temp=start;

```

```

while(temp!=NULL)
{
    printf("%d\t",temp->data);
    temp = temp->link;
}
}

```

```

void delete_head(){
    node *ptr;
    ptr = start;
    start=start->link;
    free(ptr);
}

```

```

void delete_last(){
    node *ptr,*prevptr;
    ptr = start;
    prevptr = start;
    while(ptr->link != NULL)
    {
        prevptr = ptr;
        ptr = ptr->link;
    }
    prevptr->link = NULL;
    free(ptr);
}

```

```

void delete_val(){
    int val;
    node *ptr,*prevptr;
    prevptr = start;
    ptr = start;

```

```

    printf("enter value to be deleted");
    scanf("%d",&val);
    while(ptr->data!=val){
        prevptr = ptr;
        ptr = ptr->link;
    }
    prevptr->link = ptr->link;
    free(ptr);
}

```

Output:

```

1.Create
2.Display
3.Delete Head
4.Delete Last
5.Delete val
6.Exit
Enter your choice:
1
Enter element
10
Do you want to add another element(1/0)
1
Enter element
20
Do you want to add another element(1/0)
1
Enter element
30
Do you want to add another element(1/0)
1
Enter element
40
Do you want to add another element(1/0)
0

1.Create
2.Display
3.Delete Head
4.Delete Last
5.Delete val
6.Exit
Enter your choice:
3

1.Create
2.Display
3.Delete Head
4.Delete Last
5.Delete val
6.Exit
Enter your choice:
2
20      30      40

```

```
1.Create
2.Display
3.Delete Head
4.Delete Last
5.Delete val
6.Exit
Enter your choice:
5
enter value to be deleted30
```

```
1.Create
2.Display
3.Delete Head
4.Delete Last
5.Delete val
6.Exit
Enter your choice:
2
20      40
```

```
1.Create
2.Display
3.Delete Head
4.Delete Last
5.Delete val
6.Exit
Enter your choice:
4
```

```
1.Create
2.Display
3.Delete Head
4.Delete Last
5.Delete val
6.Exit
Enter your choice:
2
20
```

```
1.Create
2.Display
3.Delete Head
4.Delete Last
5.Delete val
6.Exit
Enter your choice:
6
```

## **LAB PROGRAM 8: Implementation of Singly Linked List with sorting, reversing and concatenation of two linked lists**

WAP to Implement Single Link List with following operations

- a) Sort the linked list.
- b) Reverse the linked list.
- c) Concatenation of two linked lists

Program:

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

struct NODE
{
    int data;
    struct NODE *link;
};

typedef struct NODE node;
node *start = NULL,*start1,*start2,*start3;

node* create()
{
    int choice;
    node *new, *curr;
    start = (node*)malloc(sizeof(node));
    curr = start;
    printf("Enter element:\n");
    scanf("%d", &start->data);
    while(1)
    {
        printf("Do you want to add an element? press 1 for yes\n");
        scanf("%d", &choice);
```

```

if(choice)
{
new = (node*)malloc(sizeof(node));
printf("Please enter element:\n");
scanf("%d", &new->data);
curr->link=new;
curr = new;
}
else
{
curr->link=NULL;
break;
}
}
return start;
}

```

```

void sort()
{
    int t,n,count=0,i,j;
    node *a,*b,*temp;
    temp=start;
    while(temp!=NULL)
    {
        count++;
        temp=temp->link;
    }
    n=count;
    a=start;
    b=start->link;
    for(i=0;i<n-1;i++)
    {

```

```

        for(j=0;j<n-i-1;j++)
        {
            if(a->data>b->data)
            {
                t=a->data;
                a->data=b->data;
                b->data=t;
            }
            a=b;
            b=b->link;
        }
        a=start;
        b=start->link;
    }
}

void reverse()
{
    node*a=start,*b=NULL,*c=NULL;
    while(a!=NULL)
    {
        c=b;
        b=a;
        a=a->link;
        b->link=c;
    }
    start=b;
}

void display()
{
    node *temp;

```



```

temp = start;
if(start==NULL)
{
printf("Linked list is empty\n");
return;
}
while(temp!=NULL)
{
printf("%d\t", temp->data);
temp= temp->link;
}
}

void concatenate(node *start1,node *start2)
{
    node *temp;
    if(start1==NULL)
    {
        start=start2;
        return;
    }
    if(start2==NULL)
    {
        start=start1;
        return;
    }
    else
    {
        temp=start1;
        while(temp->link!=NULL)
        {
            temp=temp->link;

```

```

    }
    temp->link=start2;
    start=start1;
    }
}

void main()
{
    int choice,c1,c2;
    while(1)
    {
        printf("\n1.Create 2.Sort 3.Reverse 4.Concatenate 5.Display
6.Exit\n");
        printf("Enter choice:");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1: create();
            break;
            case 2: sort();
            break;
            case 3: reverse();
            break;
            case 4: printf("Do ypu want to create the first linked list if
yes press 1\n");
            scanf("%d",&c1);
            if(c1==1)
            start1=create();
            else
            start1=NULL;
            printf("Do ypu want to create the second linked list if yes press
2\n");
            scanf("%d",&c2);

```

```
        if(c2==2)
            start2=create();
        else
            start2=NULL;
        concatenate(start1,start2);
        break;
        case 5:display();
        break;
        case 6:exit(0);
        break;
        default: printf("Invalid choice\n");
    }
}
}
```

Output:

```
1.Create 2.Sort 3.Reverse 4.Concatenate 5.Display 6.Exit
Enter choice:1
Enter element:
72
Do you want to add an element? press 1 for yes
1
Please enter element:
36
Do you want to add an element? press 1 for yes
1
Please enter element:
46
Do you want to add an element? press 1 for yes
1
Please enter element:
23
Do you want to add an element? press 1 for yes
1
Please enter element:
85
Do you want to add an element? press 1 for yes
1
Please enter element:
9
Do you want to add an element? press 1 for yes
1
Please enter element:
67
Do you want to add an element? press 1 for yes
0

1.Create 2.Sort 3.Reverse 4.Concatenate 5.Display 6.Exit
Enter choice:5
72      36      46      23      85      9      67
1.Create 2.Sort 3.Reverse 4.Concatenate 5.Display 6.Exit
Enter choice:2

1.Create 2.Sort 3.Reverse 4.Concatenate 5.Display 6.Exit
Enter choice:5
9      23      36      46      67      72      85
1.Create 2.Sort 3.Reverse 4.Concatenate 5.Display 6.Exit
Enter choice:3

1.Create 2.Sort 3.Reverse 4.Concatenate 5.Display 6.Exit
Enter choice:5
85      72      67      46      36      23      9
1.Create 2.Sort 3.Reverse 4.Concatenate 5.Display 6.Exit
Enter choice:6
```

## **LAB PROGRAM 9: Implementation of Stacks and Queues using Linked Representation**

WAP to implement Stack & Queues using Linked Representation

Program:

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

struct NODE
{
    int data;
    struct NODE *link;
};

typedef struct NODE node;
node *front=NULL,*rear=NULL,*new=NULL;

void disp()
{
    node *temp;
    if(front==NULL)
    {
        printf("Empty");
        return;
    }
    temp=front;
    while(temp!=NULL)
    {
        printf("%d\t",temp->data);
        temp=temp->link;
    }
}
```

```

void ins_beg()
{
    new=(node*)malloc(sizeof(node));
    printf("Enter element:");
    scanf("%d",&new->data);
    if(front==NULL)
    {
        front=new;
        rear=new;
        new->link=NULL;
        return;
    }
    new->link=front;
    front=new;
}

```

```

void ins_end()
{
    node *temp;
    temp=rear;
    new=(node*)malloc(sizeof(node));
    printf("Enter element:");
    scanf("%d",&new->data);
    if(front==NULL)
    {
        front=new;
        rear=new;
        new->link=NULL;
        return;
    }
    new->link=NULL;
    temp->link=new;
}

```

```

        temp=temp->link;
        rear=temp;
    }

void del_beg()
{
    node *temp;
    temp=front;
    if (front==NULL)
    {
        printf("Empty");
        return;
    }
    front=front->link;
    free(temp);
}

void stackop()
{
    int c1;
    while(1)
    {
        printf("\n1.Push 2.Pop 3.Display 4.Exit");
        printf("\nEnter your choice:");
        scanf("%d",&c1);
        switch(c1)
        {
            case 1:ins_beg();
                    break;
            case 2:del_beg();
                    break;
            case 3:disp();

```

```

        break;
    case 4:exit(0);
        break;
    default:printf("Wrong choice!");
}
}

void queueop()
{
    int c1;
    while(1)
    {
        printf("\n1.Insert 2.Delete 3.Display 4.Exit");
        printf("\nEnter your choice:");
        scanf("%d",&c1);
        switch(c1)
        {
            case 1:ins_end();
                break;
            case 2:del_beg();
                break;
            case 3:disp();
                break;
            case 4:exit(0);
                break;
            default:printf("Wrong choice!");
        }
    }
}

```



```
void main()
{
    int c1;
    printf("1.Stack 2.Queue");
    printf("\nEnter your choice:");
    scanf("%d",&c1);
    switch(c1)
    {
        case 1:stackop();
            break;
        case 2:queueop();
            break;
        default:printf("Wrong Choice!");
    }
}
```

Output:

```
1.Stack 2.Queue
Enter your choice:1

1.Push 2.Pop 3.Display 4.Exit
Enter your choice:1
Enter element:10

1.Push 2.Pop 3.Display 4.Exit
Enter your choice:1
Enter element:20

1.Push 2.Pop 3.Display 4.Exit
Enter your choice:1
Enter element:30

1.Push 2.Pop 3.Display 4.Exit
Enter your choice:3
30      20      10
1.Push 2.Pop 3.Display 4.Exit
Enter your choice:2

1.Push 2.Pop 3.Display 4.Exit
Enter your choice:3
20      10
1.Push 2.Pop 3.Display 4.Exit
Enter your choice:2

1.Push 2.Pop 3.Display 4.Exit
Enter your choice:3
10
1.Push 2.Pop 3.Display 4.Exit
Enter your choice:2

1.Push 2.Pop 3.Display 4.Exit
Enter your choice:3
Empty
1.Push 2.Pop 3.Display 4.Exit
Enter your choice:4
```

```

1.Stack 2.Queue
Enter your choice:2

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:1
Enter element:10

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:1
Enter element:20

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:1
Enter element:30

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:1
Enter element:40

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:3
10      20      30      40
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:2

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:3
20      30      40
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:2

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:3
30      40
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:2

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:3
40
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:2

1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:3
Empty
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice:4

```

## **LAB PROGRAM 10: Implementation of Doubly Linked List with primitive operations**

WAP to Implement doubly link list with primitive operations

- a) Create a doubly linked list.
- b) Insert a new node to the left of the node.
- c) Delete the node based on a specific value.
- d) Display the contents of the list.

Program:

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

struct NODE
{
    struct NODE *llink;
    int data;
    struct NODE *rlink;
};

typedef struct NODE node;
node *start=NULL,*curr,*new,*temp;

void create()
{
    start=(node*)malloc(sizeof(node));
    printf("Enter element:");
    scanf("%d",&start->data);
    start->llink=NULL;
    curr=start;
    while(1)
    {
        int choice;
```

```

printf("Do you want to add an element? press 1 for yes\n");
scanf("%d", &choice);
if(choice!=0)
{
    new = (node*)malloc(sizeof(node));
    curr->rlink=new;
    new->llink=curr;
    printf("Enter the element:");
    scanf("%d",&new->data);
    curr=new;
}
else
{
    curr->rlink=NULL;
    break;
}
}
}

```

```

void insert_beg()
{
    new=(node*)malloc(sizeof(node));
    printf("Enter an element:");
    scanf("%d",&new->data);
    if(start==NULL)
    {
        new->llink=NULL;
        new->rlink=NULL;
        start=new;
        return;
    }
    new->rlink=start;
}

```

```

    start->llink=new;
    new->llink=NULL;
    start=new;
}

void delete_ele()
{
    node *temp;
    int ele;
    if(start==NULL)
    {
        printf("Linked list is empty\n");
        return;
    }
    printf("Enter element to be deleted:");
    scanf("%d",&ele);
    if(start->data==ele)
    {
        temp=start;
        start=start->rlink;
        start->llink=NULL;
        free(temp);
        return;
    }
    temp=start;
    while(temp->rlink!=NULL&&temp->data!=ele)
    {
        temp=temp->rlink;
    }
    if(temp->data==ele&&temp->rlink==NULL)
    {
        temp->llink->rlink=NULL;
    }
}

```

```

        free(temp);
        return;
    }
    if(temp->data==ele&&temp->rlink!=NULL)
    {
        temp->llink->rlink=temp->rlink;
        temp->rlink->llink=temp->llink;
        free(temp);
        return;
    }
    printf("Element not found\n");
}

void display()
{
    if(start==NULL)
    {
        printf("Linked list is empty\n");
        return;
    }
    temp=start;
    while(temp!=NULL)
    {
        printf("%d\t",temp->data);
        temp=temp->rlink;
    }
}

void main()
{
    int choice;

    printf("1.CREATE\n2.INSERT AT BEGINING\n3.DELETE SPECIFIC\n4.DISPLAY\n5.EXIT\n");

```

```

while(1)
{
    printf("Enter choice:\n");
    scanf("%d", &choice);
    switch(choice)
    {
        case 1: create();
        break;
        case 2: insert_beg();
        break;
        case 3: delete_ele();
        break;
        case 4: display();
        break;
        case 5: exit(0);
        break;
        default: printf("Invalid choice\n");
    }
}
getch();
}

```



Output:

```
1.CREATE
2.INSERT AT BEGINING
3.DELETE SPECIFIC ELEMENT
4.DISPLAY
5.EXIT
Enter choice:
1
Enter element:10
Do you want to add an element? press 1 for yes
1
Enter the element:20
Do you want to add an element? press 1 for yes
1
Enter the element:30
Do you want to add an element? press 1 for yes
1
Enter the element:40
Do you want to add an element? press 1 for yes
0
Enter choice:
4
10      20      30      40      Enter choice:
2
Enter an element:5
Enter choice:
4
5      10      20      30      40      Enter choice:
3
Enter element to be deleted:20
Enter choice:
4
5      10      30      40      Enter choice:
5
```