Experiment No 5

Name:- Anuj Rajendra Mane

ROll No:- 65

Div:-A

Subject:- Data Structures

Title: Implementation of Linked List

Problem Statements:

1) Write a program to implement following operations on Singly Linked List.

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

struct node
{
   int data;
   struct node *next;
} * start, *tail, *t, *p, *q, *n;

void create()
{
   n = (struct node *)malloc(sizeof(struct node));
   int x;
   printf("Enter the Data : ");
   scanf("%d", &x);
   n->data = x;
   n->next = 0;
}
void add_beg()
{
   create();
   if (start == 0)
   {
      start = n;
      tail = n;
   }
   else
   {
      n->next = start;
      start = n;
   }
}

void add_end()
{
   create();
   if (start == 0)
   {
      start = n;
      tail = n;
   }
   else
   {
      tail->next = n;
      tail = n;
   }
}
void display()
{
   if (start == 0)
```

```
    {
        printf("List is EMPTY\n");
    }
    else
    {
        t = start;
        while (t != 0)
        {
            printf("%u  %d  %u\n", t, t->data, t->next);
            t = t->next;
        }
    }
}
void del_beg()
{
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        t = start;
        start = start->next;
        free(t);
    }
}

void del_end()
{
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else if (start->next == 0)
    {
        free(start);
        start = 0;
        tail = 0;
    }
    else
    {
        t = start;
        while (t->next != tail)
        {
            t = t->next;
        }
        free(tail);
        tail = t;
        tail->next = 0;
    }
}


void add_inBet()
{
    int x;
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        create();
        printf("Enter the Data after which you want
to add the node : ");
        scanf("%d", &x);
        t = start;
        while (t->data != x && t != 0)
        {
            t = t->next;
        }
```

```
        if (t == 0)
        {
            printf("Entered Node is not present\n");
        }
        else
        {
            p = t->next;
            n->next = p;
            t->next = n;
        }
    }
}
void del_inBet()
{
    int x;
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        printf("Enter the Data of the node want to
delete : ");
        scanf("%d", &x);
        if (start->data == x)
        {
            del_beg();
        }
        else
        {
            t = start;
            while (t->next->data != x && t != 0)
            {
                t = t->next;
            }
            if (t == 0)
            {
                printf("Entered Node is not present\n");
            }
            else
            {
                q = t->next;
                t->next = q->next;
                free(q);
            }
        }
    }
}
void count()
{
    int count = 0;
    if (start == 0)
    {
        printf("List is empty");
    }

    else
    {
        t = start;
        while (t != 0)
        {
            count++;
            t = t->next;
        }
    }
    printf("Total No. of Nodes are: %d\n", count);
}
void search()
{
    int ele, count = 0;
```

```
    if (start == 0)
    {
        printf("List is empty");
    }
    else
    {
        t = start;
        printf("Enter ELement to search\n");
        scanf("%d", &ele);
        while (t != 0)
        {
            count++;
            if (t->data == ele)
            {
                printf("%d Found at location: %d\n", ele,
count);
                break;
            }
            t = t->next;
        }
        if (t == 0)
        {
            printf("Entered Node is not present\n");
        }
    }
}
int main()
{
    int ch;
    while (1)
    {
        printf("1.Display\t2.Add at Beginning\t3.Add
at Ending\t4.Delete Beginning\t5.Delete
Ending\t6.Add in Between\t7.Delete in
Between\t8.Count No. of Nodes\t9.Search Any
Element location\t10.Exit\n");
        printf("Enter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            display();
            break;
        case 2:
            add_beg();
            break;
        case 3:
            add_end();
            break;
        case 4:
            del_beg();
            break;
        case 5:
            del_end();
            break;
        case 6:
            add_inBet();
            break;
        case 7:
            del_inBet();
            break;
        case 8:
            count();
            break;
        case 9:
            search();
            break;
        case 10:
            exit(0);
            break;
        default:
```

```
            printf("Invalid Choice\n");
        }
    }
    return 0;
}
```

2) Write a program to implement all the above operations on Doubly Linked List.

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

struct node
{
    int data;
    struct node *next, *prev;
} * start, *tail, *t, *p, *q, *n;

void create()
{
    n = (struct node *)malloc(sizeof(struct node));
    int x;
    printf("Enter the Data : ");
    scanf("%d", &x);
    n->data = x;
    n->next = 0;
    n->prev = 0;
}

void display()
{
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        t = start;
        while (t != 0)
        {
            printf("%u  %d  %u\n", t->prev, t->data, t->next);
            t = t->next;
        }
    }
}

void add_beg()
{
    create();
    if (start == 0)
    {
        start = n;
        tail = n;
    }
    else
    {
        n->next = start;
        start->prev = n;
        start = n;
    }
}

void add_end()
{
    create();
    if (start == 0)
    {
        start = n;
        tail = n;
    }
```

```
    else
    {
        tail->next = n;
        n->prev = tail;
        tail = n;
    }
}

void del_beg()
{
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        t = start;
        start = start->next;
        start->prev = 0;
        free(t);
    }
}

void del_end()
{
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        t = tail->prev;
        free(tail);
        tail = t;
        tail->next = 0;
        if (t == 0)
            start = 0;
    }
}

void add_inBet()
{
    int x;
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        create();
        printf("Enter the Data after which you want
to add the node : ");
        scanf("%d", &x);
        t = start;
        while (t->data != x && t != 0)
        {
            t = t->next;
        }
        if (t == 0)
        {
            printf("Entered Node is not present\n");
        }
        else
        {
            p = t->next;
            n->next = p;
            p->prev = n;
            t->next = n;
            n->prev = t;
        }
```

```
    }
}
void del_inBet()
{
    int x;
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        printf("Enter the Data of the node want to
delete : ");
        scanf("%d", &x);
        if (start->data == x)
        {
            del_beg();
        }
        else
        {
            t = start;
            while (t->data != x && t != 0)
            {
                t = t->next;
            }
            if (t == 0)
            {
                printf("Entered Node is not present\n");
            }
            else
            {
                p = t->prev;
                q = t->next;
                p->next = q;
                q->prev = p;
                free(t);
            }
        }
    }
}
void count()
{
    int count = 0;
    if (start == 0)
    {
        printf("List is empty");
    }

    else
    {
        t = start;
        while (t != 0)
        {
            count++;
            t = t->next;
        }
    }
    printf("Total No. of Nodes are: %d\n", count);
}
void search()
{
    int ele, count = 0;
    if (start == 0)
    {
        printf("List is empty");
    }
    else
    {
        t = start;
        printf("Enter ELement to search\n");
```

```
        scanf("%d", &ele);
        while (t != 0)
        {
            count++;
            if (t->data == ele)
            {
                printf("%d Found at location: %d\n", ele,
count);
                break;
            }
            t = t->next;
        }
        if (t == 0)
        {
            printf("Entered Node is not present\n");
        }
    }
}
int main()
{
    int ch;
    while (1)
    {
        printf("1.Display\t2.Add at Beginning\t3.Add
at Ending\t4.Delete Beginning\t5.Delete
Ending\t6.Add in Between\t7.Delete in
Between\t8.Count No. of Nodes\t9.Search Any
Element location\t10.Exit\n");
        printf("Enter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            display();
            break;
        case 2:
            add_beg();
            break;
        case 3:
            add_end();
            break;
        case 4:
            del_beg();
            break;
        case 5:
            del_end();
            break;
        case 6:
            add_inBet();
            break;
        case 7:
            del_inBet();
            break;
        case 8:
            count();
            break;
        case 9:
            search();
            break;
        case 10:
            exit(0);
            break;
        default:
            printf("Invalid Choice\n");
        }
    }
    return 0;
}
```

3) Write a program to implement Circular Linked List.

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

struct node
{
   int data;
   struct node *next;
} * start, *tail, *t, *p, *q, *n;

void create()
{
   n = (struct node *)malloc(sizeof(struct node));
   int x;
   printf("Enter the Data : ");
   scanf("%d", &x);
   n->data = x;
   n->next = 0;
}
void add_beg()
{
   create();
   if (start == 0)
   {
      start = n;
      n->next = start;
      tail = n;
   }
   else
   {
      n->next = start;
      start = n;
      tail->next = start;
   }
}
void add_end()
{
   create();
   if (start == 0)
   {
      start = n;
      n->next = start;
      tail = n;
   }
   else
   {
      tail->next = n;
      n->next = start;
      tail = n;
   }
}
void display()
{
   if (start == 0)
   {
      printf("List is EMPTY\n");
   }
   else
   {
      t = start;
      do
      {
         printf("%u %d %u\n", t, t->data, t->next);
         t = t->next;
      } while (t != start);
   }
}
void del_beg()
{
   if (start == 0)
```

```
    {
        printf("List is EMPTY\n");
    }
    else
    {
        t = start;
        start = start->next;
        free(t);
        tail->next = start;
    }
}

void del_end()
{
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        t = start;
        while (t->next != tail)
        {
            t = t->next;
        }
        free(tail);
        tail = t;
        tail->next = start;
    }
}

void add_inBet()
{
    int x;
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
        create();
        printf("Enter the Data after which you want to add the node : ");
        scanf("%d", &x);
        t = start;
        while (t->data != x && t != 0)
        {
            t = t->next;
        }
        if (t == 0)
        {
            printf("Entered Node is not present\n");
        }
        else
        {
            p = t->next;
            n->next = p;
            t->next = n;
        }
    }
}
void del_inBet()
{
    int x;
    if (start == 0)
    {
        printf("List is EMPTY\n");
    }
    else
    {
```

```
        printf("Enter the Data of the node want to
delete : ");
        scanf("%d", &x);
        if (start->data == x)
        {
            del_beg();
        }
        else
        {
            t = start;
            while (t->next->data != x && t != 0)
            {
                t = t->next;
            }
            if (t == 0)
            {
                printf("Entered Node is not present\n");
            }
            else
            {
                q = t->next;
                t->next = q->next;
                free(q);
            }
        }
    }
}
void count()
{
    int count = 0;
    if (start == 0)
    {
        printf("List is empty");
    }
    else
    {
        t = start;
        do
        {
            count++;
            t = t->next;
        } while (t != start);
    }
    printf("Total No. of Nodes are: %d\n", count);
}
void search()
{
    int ele, count = 0;
    if (start == 0)
    {
        printf("List is empty");
    }
    else
    {
        t = start;
        printf("Enter ELement to search\n");
        scanf("%d", &ele);
        do
        {
            count++;
            if (t->data == ele)
            {
                printf("%d Found at location: %d\n", ele,
count);
                break;
            }
            t = t->next;
        } while (t != start);
```

```
        if (t == start)
        {
            printf("Entered Node is not present\n");
        }
    }
}
int main()
{
    int ch;
    while (1)
    {
        printf("1.Display\t2.Add at Beginning\t3.Add at Ending\t4.Delete Beginning\t5.Delete Ending\t6.Add in Between\t7.Delete in Between\t8.Count No. of Nodes\t9.Search Any Element location\t10.Exit\n");
        printf("Enter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            display();
            break;
        case 2:
            add_beg();
            break;
        case 3:
            add_end();
            break;
        case 4:
            del_beg();
            break;
        case 5:
            del_end();
            break;
        case 6:
            add_inBet();
            break;
        case 7:
            del_inBet();
            break;
        case 8:
            count();
            break;
        case 9:
            search();
            break;
        case 10:
            exit(0);
            break;
        default:
            printf("Invalid Choice\n");
        }
    }
    return 0;
}
```

4) Write a program to implement Stack using Linked List.

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

struct Node {
    int data;
    struct Node *next;
}*top,*n;

void push(int value) {

    n = (struct Node *)malloc(sizeof(struct Node));
    n->data = value;
    if (top == NULL) {
```

```
        n->next = NULL;
    } else {
        n->next = top;
    }
    top = n;
    printf("Node is Inserted\n\n");
}

int pop() {
    if (top == NULL) {
        printf("\nStack Underflow\n");
    } else {
        struct Node *temp = top;
        int temp_data = top->data;
        top = top->next;
        free(temp);
        return temp_data;
    }
}

void display() {

    if (top == NULL) {
        printf("\nStack Underflow\n");
    } else {
        printf("The stack is \n");
        struct Node *temp = top;
        while (temp->next != NULL) {
            printf("%d--->", temp->data);
            temp = temp->next;
        }
        printf("%d--->NULL\n\n", temp->data);
    }
}

int main() {
    int choice, value;
    printf("\nImplementation of Stack using Linked List\n");
    while (1) {
        printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");
        printf("\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice) {
        case 1:
            printf("\nEnter the value to insert: ");
            scanf("%d", &value);
            push(value);
            break;
        case 2:
            printf("Popped element is :%d\n", pop());
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("\nWrong Choice\n");
        }
    }
}
```

5.Write a program to implement queue using Linked List.

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

struct node {
    int data;
```

```
    struct node * next;
}*front,*rear,*n;
void enqueue(int value) {
    struct node * n;
    n = (struct node * ) malloc(sizeof(struct node));
    n - > data = value;
    n - > next = NULL;
    if ((front == NULL) && (rear == NULL)) {
        front = rear = n;
    } else {
        rear - > next = n;
        rear = n;
    }
    printf("Node is Inserted\n\n");
}

int dequeue() {
    if (front == NULL) {
        printf("\nUnderflow\n");
        return -1;
    } else {
        struct node * temp = front;
        int temp_data = front - > data;
        front = front - > next;
        free(temp);
        return temp_data;
    }
}

void display() {
    struct node * temp;
    if ((front == NULL) && (rear == NULL)) {
        printf("\nQueue is Empty\n");
    } else {
        printf("The queue is \n");
        temp = front;
        while (temp) {
            printf("%d--->", temp - > data);
            temp = temp - > next;
        }
        printf("NULL\n\n");
    }
}

int main() {
    int choice, value;
    printf("\nImplementation of Queue using Linked
List\n");
    while (choice != 4) {

printf("1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\n");
        printf("\nEnter your choice : ");
        scanf("%d", & choice);
        switch (choice) {
            case 1:
                printf("\nEnter the value to insert: ");
                scanf("%d", & value);
                enqueue(value);
                break;
            case 2:
                printf("Popped element is :%d\n", dequeue());
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
                break;
            default:
                printf("\nWrong Choice\n");
        }
    }
    return 0;
}
```