PRE LAB DISCUSSION

We have implemented two kinds of linked lists in the previous programs - singly and doubly linked lists. We will be discussing a variation of the singly linked list.

Circular linked list - a list where the last node is linked to the first node forming a loop/circle.

more data at : https://www.programiz.com/dsa/circular-linked-list

ALGORITHM

APPEND NODE TO THE LIST Step 1 : Start

```
Step 2 : accept a new node n
Step 3 : if head is null , set head as n
Step 4 : traverse i over linked list until i->next=head, then set i->next as n
Step 5 : set n->next as head
Step 6 : Stop
```

PREPEND NODE TO THE LIST Step 1 : Start

```
Step 2 : accept a new node n
Step 3 : append n to the list
Step 4 : set head as n
```

SEARCHING IN THE LIST Step 1 : Start

```
Step 2 : accept a number n to be searched for in the list
Step 3 : set i as head,flag as false and repeat steps 4 while i->next != head
Step 4 : if i->data equals n , print found and set flag as true
Step 5 : if flag = false , print not found
Step 6 : Stop
```

DELETING A NODE IN THE LIST Step 1 : Start

```
Step 2 : accept a number to be searched for in the list
Step 3 : set flag as false , cur as head and prev as NULL
Step 4 : repeat steps 5-7 while cur!=head
Step 5 : if cur->data = x , set found as true and break
Step 6 : set cur as cur->next
Step 7 : if found = false , print not found
Step 8 : else if cur = cur->next , head = null and exit
Step 9 : else if cur = head , traverse prev over the list until prev->next!=head and head=head->next
Step 10: prev->next = cur->next
Step 11: Stop
```

PROGRAM

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

struct NODE {
    int data;
    struct NODE *next;
};
struct NODE *head=NULL;

struct NODE* create_node(int n){
    struct NODE *ptr = malloc(sizeof(struct NODE));
    ptr->next=NULL;
    ptr->data=n;
    return ptr;
}

void append_list(struct NODE *n){
    if(head == NULL)
        head=n;
    else{
        struct NODE *i=head;
        do{
            i=i->next;
        }while(i->next!=head);

        i->next=n;
    }
    n->next=head;
}

void prepend_list(struct NODE *n){
    append_list(n);
    head=n;
}

bool isEmpty(){
    return !head;
}

int search_list(int x){
    int count=0;
    struct NODE *i=head;
    do{
        if(i->data==x)
            return count;
        i=i->next;
    }while(i!=head);
    return -1;
```

```
}
bool delete_list(int x){
    bool found=false;
    struct NODE *cur=head,*prev=NULL;
    do{
        if(cur->data==x){
            found=true;
            break;
        }
        prev=cur;
        cur=cur->next;
    }while(cur!=head);

    if(!found)
        return false;
    if(cur==cur->next){
        head=NULL;
        free(cur);
        return true;
    }
    else if(cur==head){
        prev=head;
        do{
            prev=prev->next;
        }while(prev->next!=head);
        head=head->next;
    }
    prev->next=cur->next;
    free(cur);
    return true;
}

void display_list(){
    if(isEmpty())
        printf("Empty List");
    else{
        struct NODE *i=head;
        do{
            printf("%d ",i->data);
            i=i->next;
        }while(i!=head);
    }
}

void free_list(){

    struct NODE *temp=NULL,*i=head;
    do{
        free(temp);
        temp=i;
        i=i->next;
    }while(i!=head);
    free(i);
}
int main(){

    int num=0,c=1;
    struct NODE *ptr;
    printf("1.Append to the list\n2.Prepend to the list\n3.Search in list\n4.Is the
↪   list empty?\n5.Delete from the list\n6. Display list\n");
    do{
        printf("\nEnter your choice : ");
        scanf("%d",&c);

        switch(c){
            case 1 : printf("Enter the number to be inserted : ");
                     scanf("%d",&num);
                     ptr=create_node(num);
                     append_list(ptr);
                     break;

            case 2 : printf("Enter the number to be inserted : ");
                     scanf("%d",&num);
                     ptr=create_node(num);
                     prepend_list(ptr);
                     break;

            case 3 : printf("Enter the number to be searched : ");
                     scanf("%d",&num);
                     int loc = search_list(num);
                     if(loc==-1)
                         printf("Number not found.");
                     else
                        printf("The number at location %d",num);
                     break;

            case 4 : if(isEmpty()==true)
                        printf("The list is empty");
                     else
                        printf("The list is not empty");
                     break;

            case 5 : printf("Enter the number to be deleted : ");
                     scanf("%d",&num);
                     if(delete_list(num))
                        printf("%d deleted successfully",num);
                     else
                        printf("Deletion failed. Number not found.");
                     break;

            case 6 : display_list();
                     break;;
        }
    }while(c!=0);
    free_list();
    return 0;
}
```

Output

```
1.Append to the list
2.Prepend to the list
3.Search in list
4.Is the list empty?
5.Delete from the list
6. Display list

Enter your choice : 1
Enter the number to be inserted : 1

Enter your choice : 1
Enter the number to be inserted : 2

Enter your choice : 1
Enter the number to be inserted : 3

Enter your choice : 6
1 2 3
Enter your choice : 2
Enter the number to be inserted : 0

Enter your choice : 6
0 1 2 3
Enter your choice : 3
Enter the number to be searched : 2
The number at location 2
Enter your choice : 6
0 1 2 3
Enter your choice : 4
The list is not empty
Enter your choice : 5
Enter the number to be deleted : 0
0 deleted successfully
Enter your choice : 6
1 2 3
Enter your choice : 5
Enter the number to be deleted : 3
3 deleted successfully
Enter your choice : 6
1 2
Enter your choice : 5
Enter the number to be deleted : 2
2 deleted successfully
Enter your choice : 5
Enter the number to be deleted : 1
```

```
1 deleted successfully
Enter your choice : 6
Empty List
Enter your choice : 0
```