**LAB SESSION 5: DOUBLY LINKED LIST AND CIRCULAR SINGLY LINKED LIST**

**Aim: To implement doubly linked list and circular singly linked list**

**Problem definition:**

* Develop a C program to implement the following

1. Implement a doubly linked list(add, delete, reverse)
2. Swap the adjacent elements of doubly linked list by rearranging the links.
3. Create a doubly linked list in which the info part of each node contains a digit of a given number. The digits should be stored in reverse order. i.e., the least significant digit should be stored in the first node and the most significant digit should be stored in the last node.Example: If number entered in 5679 the list should be 9 -> 7 ->6 -> 5. Write a function to add two numbers using the above linked list representation.
4. Delete alternate nodes from a circular singly linked list

**THEORY**:

**Doubly linked list**

Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence. Therefore, in a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer) , pointer to the previous node (previous pointer). A sample node in a doubly linked list is shown in the figure.



A doubly linked list containing three nodes having numbers from 1 to 3 in their data part, is shown in the following image.



**Circular Singly Linked List**

In a circular Singly linked list, the last node of the list contains a pointer to the first node of the list. We can have circular singly linked list as well as circular doubly linked list.

We traverse a circular singly linked list until we reach the same node where we started. The circular singly liked list has no beginning and no ending. There is no null value present in the next part of any of the nodes.

The following image shows a circular singly linked list.



Circular linked list are mostly used in task maintenance in operating systems. There are many examples where circular linked list are being used in computer science including browser surfing where a record of pages visited in the past by the user, is maintained in the form of circular linked lists and can be accessed again on clicking the previous button.

#include<stdio.h>

#include<stdlib.h>

struct node{

    struct node \*prev;

    int info;

    struct node \*next;

};

struct node \* addatbeg(struct node \*start, int data){

    struct node \*temp = (struct node\*)malloc(sizeof(struct node));

    temp->info = data;

    temp->next = start->next;

    temp->prev = NULL;

    start = temp;

    return start;

}

struct node \* addatend(struct node \*start, int data){

    struct node \*p = start, \*temp;

    temp = (struct node\*)malloc(sizeof(struct node));

    temp->info = data;

    temp->next = NULL;

    while(p->next != NULL)

        p = p->next;

    temp->prev = p;

    p->next = temp;

    return start;

}

struct node \* create(struct node \*start){

    int n, data;

    printf("Enter number of nodes: ");

    scanf("%d", &n);

    if(n == 0)

        return start;

    printf("Enter data: ");

    scanf("%d", &data);

    start = addtoempty(start, data);

    for(int i = 1; i < n; i++){

        printf("Enter data: ");

        scanf("%d", &data);

        start = addatend(start, data);

    }

    return start;

}

struct node \* addbefore(struct node \*start, int data, int item){

    struct node \*p = start->next, \*temp;

    if(start->info == item){

        temp = (struct node\*)malloc(sizeof(struct node));

        temp->info = data;

        temp->next = start->next;

        temp->prev = NULL;

        start = temp;

        return start;

    }

    while(p != NULL){

        if(p->info == item){

            temp = (struct node\*)malloc(sizeof(struct node));

            temp->info = data;

            temp->prev = p->prev;

            temp->next = p;

            p->prev->next = temp;

            p->prev = temp;

            return start;

        }

        p = p->next;

    }

    printf("Item not found\n");

    return start;

}

struct node \* addafter(struct node \*start, int data, int item){

    struct node \*p = start, \*temp;

    temp = (struct node\*)malloc(sizeof(struct node));

    while(p != NULL){

        if(p->info == item){

            temp->info = data;

            temp->prev = p;

            temp->next = p->next;

            if(p->next != NULL)

                p->next->prev = temp;

            p->next = temp;

            return start;

        }

        p = p->next;

    }

    printf("Item not found\n");

    return start;

}

struct node \* delete(struct node \*start, int item){

    struct node \*temp;

    if(start->next == NULL){

        if(start->info == item){

            temp = start;

            start = NULL;

            free(temp);

            return start;

        }

    }

    if(start->info == item){

        temp = start;

        start = start->next;

        free(temp);

        return start;

    }

    temp = start->next;

    while(temp->next != NULL){

        if(temp->info == item){

            temp->next->prev = temp->next;

            temp->prev->next = temp->prev;

            free(temp);

            return start;

        }

        temp = temp->next;

    }

    if(temp->info == item){

        temp->prev->next = NULL;

        free(temp);

        return start;

    }

    printf("Item not found\n");

    return start;

}

struct node \* reverse(struct node \*start){

    struct node \*p = start, \*q = start->next;

    p->prev = q;

    p->next = NULL;

    while(q != NULL){

        q->prev = q->next;

        q->next = p;

        p = q;

        q = q->prev;

    }

    start = p;

    return start;

}

void display(struct node \*start){

    struct node \*p = start;

    if(start == NULL){

        printf("Empty list\n");

        return;

    }

    printf("List contents\n");

    do{

        printf(" %d", p->info);

        p = p->next;

    }while(p != NULL);

}

int main(){

    struct node \*start = NULL;

    int s, data, item;

    do{

        printf("\n1. Creating a list");

        printf("\n2. Add at beginning");

        printf("\n3. Add at end");

        printf("\n4. Add before a node");

        printf("\n5. Add after a node");

        printf("\n6. Delete a node");

        printf("\n7. Reverse the list");

        printf("\n8. Display the list");

        printf("\n9. Exit");

        printf("\nEnter your option: ");

        scanf("%d", &s);

        switch(s){

            case 1: start = create(start);

                    display(start);

                    break;

            case 2: printf("Enter the data: ");

                    scanf("%d", &data);

                    start = addatbeg(start, data);

                    display(start);

                    break;

            case 3: printf("Enter the data: ");

                    scanf("%d", &data);

                    start = addatend(start, data);

                    display(start);

                    break;

            case 4: printf("Enter the data and item: ");

                    scanf("%d %d", &data, &item);

                    start = addbefore(start, data, item);

                    display(start);

                    break;

            case 5: printf("Enter the data and item: ");

                    scanf("%d %d", &data, &item);

                    start = addafter(start, data, item);

                    display(start);

                    break;

            case 6: printf("Enter the item: ");

                    scanf("%d",&item);

                    start = delete(start, item);

                    display(start);

                    break;

            case 7: start = reverse(start);

                    printf("The list is reversed\n");

                    display(start);

                    break;

            case 8: display(start);

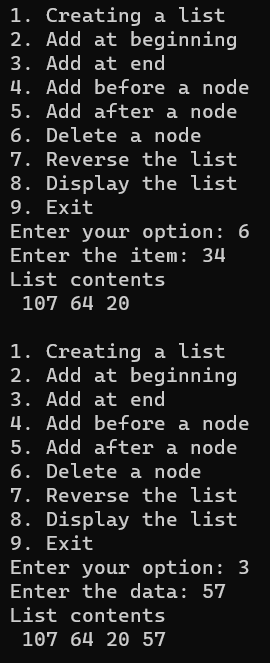
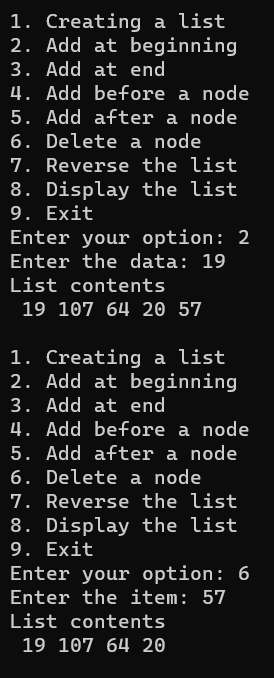
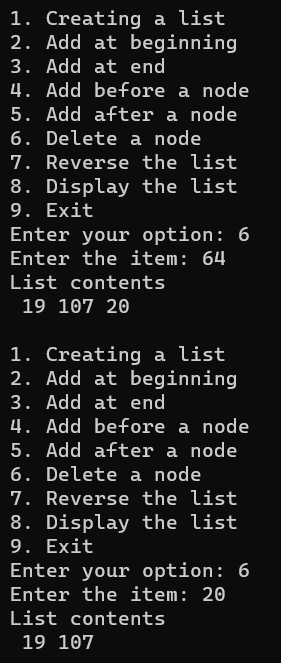
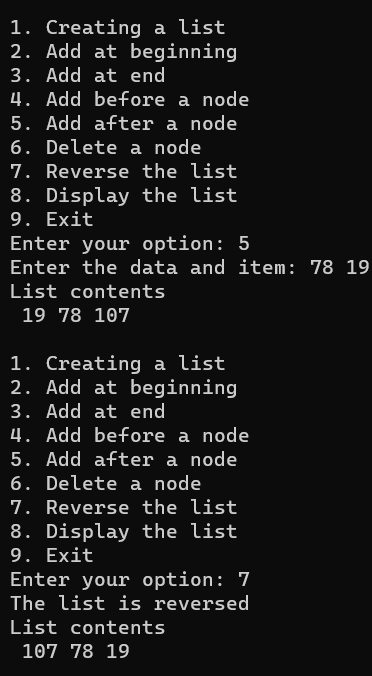
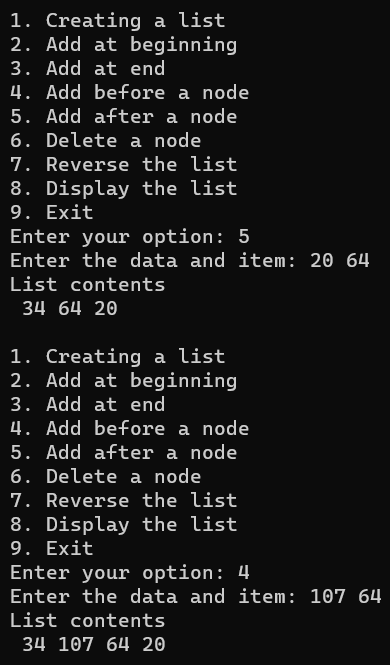
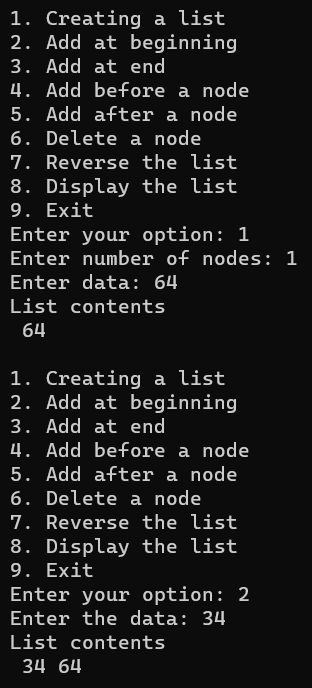
                    break;

            case 9: break;

            default: printf("Invalid input\n");

        }

    }while(s != 9); }



#include<stdio.h>

#include<stdlib.h>

struct node{

    struct node \*prev;

    int info;

    struct node \*next;

};

struct node \* addtoempty(struct node \*, int);

struct node \* addatend(struct node \*, int);

struct node \* create(struct node \*);

struct node \* swap(struct node \*);

void display(struct node\*);

struct node \* addtoempty(struct node \*start, int data){

    struct node \*temp = (struct node\*)malloc(sizeof(struct node));

    temp->info = data;

    temp->next = NULL;

    temp->prev = NULL;

    start = temp;

    return start;

}

struct node \* addatend(struct node \*start, int data){

    struct node \*p = start, \*temp;

    temp = (struct node\*)malloc(sizeof(struct node));

    temp->info = data;

    temp->next = NULL;

    while(p->next != NULL)

        p = p->next;

    temp->prev = p;

    p->next = temp;

    return start;

}

struct node \* create(struct node \*start){

    int n, data;

    printf("Enter number of nodes: ");

    scanf("%d", &n);

    if(n == 0)

        return start;

    printf("Enter data: ");

    scanf("%d", &data);

    start = addtoempty(start, data);

    for(int i = 1; i < n; i++){

        printf("Enter data: ");

        scanf("%d", &data);

        start = addatend(start, data);

    }

    return start;

}

struct node \* swap(struct node \*start) {

    struct node \*p = start, \*q = start->next;

    while (p != NULL && q != NULL) {

        p->next = q->next;

        q->prev = p->prev;

        p->prev = q;

        q->next = p;

        if (p->next != NULL) {

            p->next->prev = p;

        }

        if (q->prev != NULL) {

            q->prev->next = q;

        }

        if (start == p) {

            start = q;

        }

        p = p->next;

        if (p != NULL) {

            q = p->next;

        }

    }

    return start;

}

void display(struct node \*start){

    struct node \*p = start;

    if(start == NULL){

        printf("Empty list\n");

        return;

    }

    printf("List contents\n");

    do{

        printf(" %d", p->info);

        p = p->next;

    }while(p != NULL);

    printf("\n");

}

int main(){

    struct node \*start = NULL;

    printf("Fill the list\n");

    start = create(start);

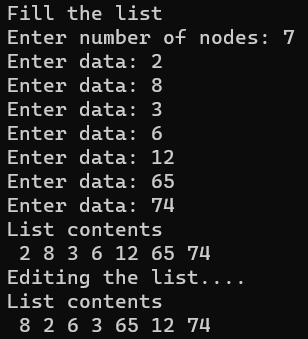
    display(start);

    printf("Editing the list....\n");

    start = swap(start);

    display(start);

}



#include <stdio.h>

#include <stdlib.h>

struct node {

    struct node\* prev;

    int info;

    struct node\* next;

};

struct node\* addtoempty(struct node\* start, int data) {

    struct node\* temp = (struct node\*)malloc(sizeof(struct node));

    temp->info = data;

    temp->next = NULL;

    temp->prev = NULL;

    start = temp;

    return start;

}

struct node\* addatend(struct node\* start, int data) {

    struct node\* p = start, \* temp;

    temp = (struct node\*)malloc(sizeof(struct node));

    temp->info = data;

    temp->next = NULL;

    if (start == NULL) {

        temp->prev = NULL;

        start = temp;

        return start;

    }

    while (p->next != NULL)

        p = p->next;

    temp->prev = p;

    p->next = temp;

    return start;

}

struct node\* create(struct node\* start, int n) {

    int data;

    if (n == 0)

        return (addtoempty(start, 0));

    while (n > 0) {

        data = n % 10;

        n = n / 10;

        start = addatend(start, data);

    }

    return start;

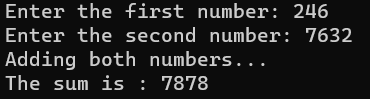
}

struct node\* add(struct node\* num1, struct node\* num2) {

    struct node\* p = num1, \* q = num2, \* sum = NULL;

    int s = 0, carry = 0;

    while (p != NULL || q != NULL) {

        int x = (p != NULL) ? p->info : 0;

        int y = (q != NULL) ? q->info : 0;

        s = x + y + carry

        carry = s / 10;

        s = s % 10;

        sum = addatend(sum, s);

        if (p != NULL) p = p->next;

        if (q != NULL) q = q->next;

    }

    if (carry > 0) {

        sum = addatend(sum, carry);

    }

    return sum;

}

struct node\* reverse(struct node\* start) {

    struct node\* p = start, \* temp = NULL;

    while (p != NULL) {

        temp = p->prev;

        p->prev = p->next;

        p->next = temp;

        p = p->prev;

    }

    return start;

}

void display(struct node\* start) {

    struct node\* p = start;

    if (start == NULL) {

        printf("0\n");

        return;

    }

    do {

        printf("%d", p->info);

        p = p->next;

    } while (p != NULL);

    printf("\n");

}

int main() {

    struct node\* num1 = NULL, \* num2 = NULL, \* sum = NULL;

    int n1, n2;

    printf("Enter the first number: ");

    scanf("%d", &n1);

    num1 = create(num1, n1);

    printf("Enter the second number: ");

    scanf("%d", &n2);

    num2 = create(num2, n2);

    printf("Adding both numbers...\n");

    sum = add(num1, num2);

    sum = reverse(sum);

    printf("The sum is : ");

    display(sum);

    return 0;

}

#include<stdio.h>

#include<stdlib.h>

struct node{

    int info;

    struct node \*link;

};

struct node \* addtoempty(struct node \*, int);

struct node \* addatend(struct node \*, int);

struct node \* create(struct node \*);

struct node \* delete(struct node \*);

void display(struct node\*);

struct node \* addtoempty(struct node \*last, int data){

    struct node \*temp = (struct node\*)malloc(sizeof(struct node));

    temp->info = data;

    last = temp;

    last->link = last;

    return last;

}

struct node \* addatend(struct node \*last, int data){

    struct node \*temp;

    temp = (struct node\*)malloc(sizeof(struct node));

    temp->info = data;

    temp->link = last->link;

    last->link = temp;

    last = temp;

    return last;

}

struct node \* create(struct node \*last){

    int n, data;

    printf("Enter number of nodes: ");

    scanf("%d", &n);

    if(n == 0)

        return last;

    printf("Enter data: ");

    scanf("%d", &data);

    last = addtoempty(last, data);

    for(int i = 1; i < n; i++){

        printf("Enter data: ");

        scanf("%d", &data);

        last = addatend(last, data);

    }

    return last;

}

struct node \* delete(struct node \*last){

    if (last == NULL) {

        printf("Empty list\n");

        exit(1);

    }

    struct node\* p = last;

    struct node\* temp;

    do {

        temp = p->link;

        p->link = temp->link;

        free(temp);

        p = p->link;

    } while (p!=last && p->link!=last);

    return last;

}

void display(struct node \*last){

    struct node \*p = last->link;

    if(last == NULL){

        printf("Empty list\n");

        return;

    }

    printf("List contents: ");

    do{

        printf(" %d", p->info);

        p = p->link;

    }while(p != last->link);

}

int main(){

    struct node \*last = NULL;

    printf("Fill the list\n");

    last = create(last);

    printf("Editing the list...\n");

    last = delete(last);

    display(last);

}

