DS Lab Assignment 2

Pranav Joshi
CS-B
Roll no: 43

Title: WAP to Implement following operations on SLL. a. Create b. Insert c. Delete d. Display e. Reverse

Code:

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

struct Node {
    int data;
    struct Node* next;
};

struct Node* createLinkedList(int n);
void displayLinkedList(struct Node* head);
struct Node* insertNode(struct Node* head, int value, int position);
struct Node* deleteNode(struct Node* head, int position);
struct Node* reverseLinkedList(struct Node* head);

int main() {
    int n, value, position;
    struct Node* head = NULL;

    printf("Enter the number of nodes: ");
    scanf("%d", &n);

    head = createLinkedList(n);

    printf("The entered linked list is: ");
    displayLinkedList(head);

    printf("Enter the value and position to insert a node: ");
    scanf("%d %d", &value, &position);
    head = insertNode(head, value, position);

    printf("The new linked list is: ");
    displayLinkedList(head);
```

```
    printf("Enter the position of node to delete: ");
    scanf("%d", &position);
    head = deleteNode(head, position);

    printf("The linked list is: ");
    displayLinkedList(head);

    head = reverseLinkedList(head);
    printf("The reversed linked list is: ");
    displayLinkedList(head);

    return 0;
}

struct Node* createLinkedList(int n) {
    int i, value;
    struct Node* head = NULL;
    struct Node* temp = NULL;
    struct Node* p = NULL;

    for (i = 0; i < n; i++) {
        printf("Enter the value for node %d: ", i + 1);
        scanf("%d", &value);

        temp = (struct Node*)malloc(sizeof(struct Node));
        temp->data = value;
        temp->next = NULL;

        if (head == NULL) {
            head = temp;
        } else {
            p = head;
            while (p->next != NULL) {
                p = p->next;
            }
            p->next = temp;
        }
    }
    return head;
}

void displayLinkedList(struct Node* head) {
    struct Node* p = head;

    while (p != NULL) {
        printf("%d ", p->data);
        p = p->next;
    }
```

```
    printf("\n");
}

struct Node* insertNode(struct Node* head, int value, int position) {
    struct Node* temp = NULL;
    struct Node* p = head;
    int i;

    temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = value;
    temp->next = NULL;

    if (position == 1) {
        temp->next = head;
        head = temp;
    } else {
        for (i = 1; i < position - 1 && p != NULL; i++) {
            p = p->next;
        }

        if (p == NULL) {
            printf("Invalid position\n");
        } else {
            temp->next = p->next;
            p->next = temp;
        }
    }
    return head;
}

struct Node* deleteNode(struct Node* head, int position) {
    struct Node* p = head;
    struct Node* q = NULL;
    int i;

    if (position == 1) {
        head = head->next;
        free(p);
    } else {
        for (i = 1; i < position && p != NULL; i++) {
            q = p;
            p = p->next;
        }

        if (p == NULL) {
            printf("Invalid position\n");
        } else {
            q->next = p->next;
```

```
            free(p);
        }
    }
    return head;
}

struct Node * reverseLinkedList(struct Node* head) {
    struct Node *prevNode, *curNode;

    if (head == NULL) {
        return NULL;
    }

    prevNode = head;
    curNode = head->next;
    head = head->next;

    prevNode->next = NULL;

    while (head != NULL) {
        head = head->next;
        curNode->next = prevNode;

        prevNode = curNode;
        curNode = head;
    }

    head = prevNode;

    return head;
}
```

Output: