// DS LAB: 8 – SINGLE LINKED LIST

#include <stdio.h>

#include <stdlib.h>

// Define the structure for a node in the linked list

struct Node {

int data;

struct Node\* next;

};

// Function to create a new node

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Memory allocation failed\n");

exit(1);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

// Function to display the linked list

void display(struct Node\* head) {

struct Node\* current = head;

if (current == NULL) {

printf("The list is empty.\n");

return;

}

printf("Linked List: ");

while (current != NULL) {

printf("%d -> ", current->data);

current = current->next;

}

printf("NULL\n");

}

// Function to insert a node at the beginning of the linked list

struct Node\* insertAtBeginning(struct Node\* head, int data) {

struct Node\* newNode = createNode(data);

newNode->next = head;

return newNode;

}

// Function to insert a node at the middle of the linked list

void insertAtMiddle(struct Node\* prevNode, int data) {

if (prevNode == NULL) {

printf("Previous node cannot be NULL for inserting in the middle.\n");

return;

}

struct Node\* newNode = createNode(data);

newNode->next = prevNode->next;

prevNode->next = newNode;

}

// Function to insert a node at the end of the linked list

void insertAtEnd(struct Node\* head, int data) {

struct Node\* newNode = createNode(data);

struct Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

// Function to delete a node from the head of the linked list

struct Node\* deleteAtHead(struct Node\* head) {

if (head == NULL) {

printf("The list is empty. Nothing to delete.\n");

return NULL;

}

struct Node\* temp = head->next;

free(head);

return temp;

}

// Function to delete a node from a non-head position in the linked list

void deleteAtPosition(struct Node\* prevNode) {

if (prevNode == NULL || prevNode->next == NULL) {

printf("Previous node cannot be NULL for deletion.\n");

return;

}

struct Node\* temp = prevNode->next;

prevNode->next = temp->next;

free(temp);

}

int main() {

struct Node\* head = NULL;

int choice, data;

while (1) {

printf("\nLinked List Menu:\n");

printf("1. Insert at beginning\n");

printf("2. Insert at middle\n");

printf("3. Insert at end\n");

printf("4. Delete at head\n");

printf("5. Delete at non-head position\n");

printf("6. Display\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data to insert at the beginning: ");

scanf("%d", &data);

head = insertAtBeginning(head, data);

break;

case 2:

printf("Enter data to insert in the middle: ");

scanf("%d", &data);

insertAtMiddle(head, data);

break;

case 3:

printf("Enter data to insert at the end: ");

scanf("%d", &data);

if (head == NULL) {

head = createNode(data);

} else {

insertAtEnd(head, data);

}

break;

case 4:

head = deleteAtHead(head);

break;

case 5:

printf("Enter data after which to delete: ");

scanf("%d", &data);

struct Node\* current = head;

while (current != NULL && current->data != data) {

current = current->next;

}

deleteAtPosition(current);

break;

case 6:

display(head);

break;

case 7:

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

/\* Output :

Linked List Menu:

1. Insert at beginning

2. Insert at middle

3. Insert at end

4. Delete at head

5. Delete at non-head position

6. Display

7. Exit

Enter your choice: 1

Enter data to insert at the beginning: 10

Linked List Menu:

1. Insert at beginning

2. Insert at middle

3. Insert at end

4. Delete at head

5. Delete at non-head position

6. Display

7. Exit

Enter your choice: 1

Enter data to insert at the beginning: 20

Linked List Menu:

1. Insert at beginning

2. Insert at middle

3. Insert at end

4. Delete at head

5. Delete at non-head position

6. Display

7. Exit

Enter your choice: 1

Enter data to insert at the beginning: 30

Linked List Menu:

1. Insert at beginning

2. Insert at middle

3. Insert at end

4. Delete at head

5. Delete at non-head position

6. Display

7. Exit

Enter your choice: 2

Enter data to insert in the middle: 50

Linked List Menu:

1. Insert at beginning

2. Insert at middle

3. Insert at end

4. Delete at head

5. Delete at non-head position

6. Display

7. Exit

Enter your choice: 3

Enter data to insert at the end: 60

Linked List Menu:

1. Insert at beginning

2. Insert at middle

3. Insert at end

4. Delete at head

5. Delete at non-head position

6. Display

7. Exit

Enter your choice: 4

Linked List Menu:

1. Insert at beginning

2. Insert at middle

3. Insert at end

4. Delete at head

5. Delete at non-head position

6. Display

7. Exit

Enter your choice: 5

Enter data after which to delete: 10

Linked List Menu:

1. Insert at beginning

2. Insert at middle

3. Insert at end

4. Delete at head

5. Delete at non-head position

6. Display

7. Exit

Enter your choice: 6

Linked List: 50 -> 20 -> 10 -> NULL

\*/