#include <stdio.h>

#include <stdlib.h>

// Node structure

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));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void insertInMiddle(struct Node\*\* head\_ref, int data, int position) {

struct Node\* newNode = createNode(data);

if (position == 0) { // Insert at head

newNode->next = \*head\_ref;

\*head\_ref = newNode;

return;

}

struct Node\* temp = \*head\_ref;

for (int i = 0; temp != NULL && i < position - 1; i++) {

temp = temp->next;

}

if (temp == NULL) {

printf("Position exceeds list length. Insertion failed.\n");

return;

}

newNode->next = temp->next;

temp->next = newNode;

}

void deleteInMiddle(struct Node\*\* head\_ref, int position) {

if (\*head\_ref == NULL) {

printf("List is empty.\n");

return;

}

struct Node\* temp = \*head\_ref;

// If the head is to be deleted

if (position == 0) {

\*head\_ref = temp->next;

free(temp);

return;

}

for (int i = 0; temp != NULL && i < position - 1; i++) {

temp = temp->next;

}

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

printf("Position exceeds list length. Deletion failed.\n");

return;

}

struct Node\* nextNode = temp->next->next;

free(temp->next);

temp->next = nextNode;

}

void printList(struct Node\* head) {

while (head != NULL) {

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

head = head->next;

}

printf("NULL\n");

}

int main() {

struct Node\* head = NULL;

insertInMiddle(&head, 10, 0); // Insert at position 0

insertInMiddle(&head, 20, 1); // Insert at position 1

insertInMiddle(&head, 30, 2); // Insert at position 2

insertInMiddle(&head, 25, 2); // Insert at position 2

printf("Linked list after insertions:\n");

printList(head);

deleteInMiddle(&head, 2); // Delete the node at position 2

printf("Linked list after deletion:\n");

printList(head);

return 0;

}

OUTPUT

Linked list after insertions:

10 -> 20 -> 25 -> 30 -> NULL

Linked list after deletion:

10 -> 20 -> 30 -> NULL