***1// C Program to insert the node at the beginning of***

***// Linked List***

***#include <stdio.h>***

***#include <stdlib.h>***

***struct Node {***

***int data;***

***struct Node\* next;***

***};***

***// Function to create a new node with the given data***

***struct Node\* createNode(int new\_data)***

***{***

***struct Node\* new\_node***

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

***new\_node->data = new\_data;***

***new\_node->next = NULL;***

***return new\_node;***

***}***

***// Function to insert a new node at the beginning of the***

***// list***

***struct Node\* insertAtFront(struct Node\* head, int new\_data)***

***{***

***// Create a new node with the given data***

***struct Node\* new\_node = createNode(new\_data);***

***// Make the next of the new node point to the current***

***// head***

***new\_node->next = head;***

***// Return the new node as the new head of the list***

***return new\_node;***

***}***

***void printList(struct Node\* head)***

***{***

***struct Node\* curr = head;***

***while (curr != NULL) {***

***printf(" %d", curr->data);***

***curr = curr->next;***

***}***

***printf("\n");***

***}***

***int main()***

***{***

***// Create the linked list 2->3->4->5***

***struct Node\* head = createNode(2);***

***head->next = createNode(3);***

***head->next->next = createNode(4);***

***head->next->next->next = createNode(5);***

***// Print the original list***

***printf("Original Linked List:");***

***printList(head);***

***// Insert a new node at the front of the list***

***printf("After inserting Nodes at the front:");***

***int data = 1;***

***head = insertAtFront(head, data);***

***// Print the updated list***

***printList(head);***

***return 0;***

***}***

***2*** *// C program to delete a linked list node at a given*

*// position*

#include *<stdio.h>*

#include *<stdlib.h>*

*// Node structure for the linked list*

**struct** **Node** {

int data;

**struct** **Node**\* next;

};

**struct** **Node**\* newNode(int data)

{

**struct** **Node**\* node

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

node->data = data;

node->next = NULL;

**return** node;

}

*// Function to delete a node at a given position*

**struct** **Node**\* deleteNode(**struct** **Node**\* head, int position)

{

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

**struct** **Node**\* prev = NULL;

*// Base case if linked list is empty*

**if** (temp == NULL)

**return** head;

*// Case 1: Head is to be deleted*

**if** (position == 1) {

head = temp->next;

free(temp);

**return** head;

}

*// Case 2: Node to be deleted is in middle*

*// Traverse till given position*

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

prev = temp;

temp = temp->next;

}

*// If given position is found, delete node*

**if** (temp != NULL) {

prev->next = temp->next;

free(temp);

}

**else** {

printf("Data not present**\n**");

}

**return** head;

}

void printList(**struct** **Node**\* head)

{

**while** (head != NULL) {

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

head = head->next;

}

printf("NULL**\n**");

}

*// Driver code*

int main()

{

**struct** **Node**\* head = newNode(1);

head->next = newNode(2);

head->next->next = newNode(3);

head->next->next->next = newNode(4);

head->next->next->next->next = newNode(5);

printf("Original list: ");

printList(head);

int position = 2;

head = deleteNode(head, position);

printf("List after deletion: ");

printList(head);

*// Cleanup remaining nodes*

**while** (head != NULL) {

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

head = head->next;

free(temp);

}

**return** 0;

}

**2 *// Iterative C program to reverse a linked list***

**#include *<stdio.h>***

**struct Node {**

**int data;**

**struct Node\* next;**

**};**

***// Given the head of a list, reverse the list and return the***

***// head of reversed list***

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

***// Initialize three pointers: curr, prev and next***

**struct Node \*curr = head, \*prev = NULL, \*next;**

***// Traverse all the nodes of Linked List***

**while (curr != NULL) {**

***// Store next***

**next = curr->next;**

***// Reverse current node's next pointer***

**curr->next = prev;**

***// Move pointers one position ahead***

**prev = curr;**

**curr = next;**

**}**

***// Return the head of reversed linked list***

**return prev;**

**}**

***// This function prints the contents***

***// of the linked list starting from the head***

**void printList(struct Node\* node) {**

**while (node != NULL) {**

**printf(" %d", node->data);**

**node = node->next;**

**}**

**}**

***// Function to create a new node***

**struct Node\* createNode(int new\_data) {**

**struct Node\* new\_node**

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

**new\_node->data = new\_data;**

**new\_node->next = NULL;**

**return new\_node;**

**}**

**int main() {**

***// Create a hard-coded linked list:***

***// 1 -> 2 -> 3 -> 4 -> 5***

**struct Node\* head = createNode(1);**

**head->next = createNode(2);**

**head->next->next = createNode(3);**

**head->next->next->next = createNode(4);**

**head->next->next->next->next = createNode(5);**

***// Print the original linked list***

**printf("Given Linked list:");**

**printList(head);**

***// Function call to return the reversed list***

**head = reverseList(head);**

***// Print the reversed linked list***

**printf("\nReversed Linked List:");**

**printList(head);**

**return 0;**

**}**

3 *// Recursive C program to reverse a linked list*

#include *<stdio.h>*

**struct** **Node** {

int data;

**struct** **Node**\* next;

};

*// Given the head of a list, reverse the list and return the*

*// head of reversed list*

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

**if** (head == NULL || head->next == NULL)

**return** head;

*// reverse the rest of linked list and put the first element at the end*

**struct** **Node**\* rest = reverseList(head->next);

*// Make the current head as last node of remaining linked list*

head->next->next = head;

*// Update next of current head to NULL*

head->next = NULL;

*// Return the reversed linked list*

**return** rest;

}

*// Function to create a new node*

**struct** **Node**\* createNode(int new\_data) {

**struct** **Node**\* new\_node =

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

new\_node->data = new\_data;

new\_node->next = NULL;

**return** new\_node;

}

*// This function prints the contents*

*// of the linked list starting from the head*

void printList(**struct** **Node**\* node) {

**while** (node != NULL) {

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

node = node->next;

}

printf("**\n**");

}

*// Driver Code*

int main() {

*// Create a hard-coded linked list:*

*// 1 -> 2 -> 3 -> 4 -> 5*

**struct** **Node**\* head = createNode(1);

head->next = createNode(2);

head->next->next = createNode(3);

head->next->next->next = createNode(4);

head->next->next->next->next = createNode(5);

printf("Given Linked List:");

printList(head);

*// Function call to return the reversed list*

head = reverseList(head);

printf("Reversed Linked List:");

printList(head);

**return** 0;

}