## 16 June

## Write a C Program to perform the following operations on Triply Linked List (TLL).

## // A linked list node

## struct Node

## {

## int data;

## struct Node \*next;

## };

## /\* Given a reference (pointer to pointer) to the head of a list

## and an int, inserts a new node on the front of the list. \*/

## void push(struct Node\*\* head\_ref, int new\_data)

## {

## /\* 1. allocate node \*/

## struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));

## 

## /\* 2. put in the data \*/

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

## 

## /\* 3. Make next of new node as head \*/

## new\_node->next = (\*head\_ref);

## 

## /\* 4. move the head to point to the new node \*/

## (\*head\_ref) = new\_node;

## }

## /\* Given a node prev\_node, insert a new node after the given

## prev\_node \*/

## void insertAfter(struct Node\* prev\_node, int new\_data)

## {

## /\*1. check if the given prev\_node is NULL \*/

## if (prev\_node == NULL)

## {

## printf("the given previous node cannot be NULL");

## return;

## }

## 

## /\* 2. allocate new node \*/

## struct Node\* new\_node =(struct Node\*) malloc(sizeof(struct Node));

## 

## /\* 3. put in the data \*/

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

## 

## /\* 4. Make next of new node as next of prev\_node \*/

## new\_node->next = prev\_node->next;

## 

## /\* 5. move the next of prev\_node as new\_node \*/

## prev\_node->next = new\_node;

## }

## /\* Given a reference (pointer to pointer) to the head

## of a list and an int, appends a new node at the end \*/

## void append(struct Node\*\* head\_ref, int new\_data)

## {

## /\* 1. allocate node \*/

## struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));

## 

## struct Node \*last = \*head\_ref; /\* used in step 5\*/

## 

## /\* 2. put in the data \*/

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

## 

## /\* 3. This new node is going to be the last node, so make next

## of it as NULL\*/

## new\_node->next = NULL;

## 

## /\* 4. If the Linked List is empty, then make the new node as head \*/

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

## {

## \*head\_ref = new\_node;

## return;

## }

## 

## /\* 5. Else traverse till the last node \*/

## while (last->next != NULL)

## last = last->next;

## 

## /\* 6. Change the next of last node \*/

## last->next = new\_node;

## return;

## }

## // C program to delete N nodes after M nodes of a linked list

## #include <stdio.h>

## #include <stdlib.h>

## 

## // A linked list node

## struct Node

## {

## int data;

## struct Node \*next;

## };

## 

## /\* Function to insert a node at the beginning \*/

## void push(struct Node \*\* head\_ref, int new\_data)

## {

## /\* allocate node \*/

## struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));

## 

## /\* put in the data \*/

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

## 

## /\* link the old list off the new node \*/

## new\_node->next = (\*head\_ref);

## 

## /\* move the head to point to the new node \*/

## (\*head\_ref) = new\_node;

## }

## 

## /\* Function to print linked list \*/

## void printList(struct Node \*head)

## {

## struct Node \*temp = head;

## while (temp != NULL)

## {

## printf("%d ", temp->data);

## temp = temp->next;

## }

## printf("\n");

## }

## 

## // Function to skip M nodes and then delete N nodes of the linked list.

## void skipMdeleteN(struct Node \*head, int M, int N)

## {

## struct Node \*curr = head, \*t;

## int count;

## 

## // The main loop that traverses through the whole list

## while (curr)

## {

## // Skip M nodes

## for (count = 1; count<M && curr!= NULL; count++)

## curr = curr->next;

## 

## // If we reached end of list, then return

## if (curr == NULL)

## return;

## 

## // Start from next node and delete N nodes

## t = curr->next;

## for (count = 1; count<=N && t!= NULL; count++)

## {

## struct Node \*temp = t;

## t = t->next;

## free(temp);

## }

## curr->next = t; // Link the previous list with remaining nodes

## 

## // Set current pointer for next iteration

## curr = t;

## }

## }

## 

## // Driver program to test above functions

## int main()

## {

## /\* Create following linked list

## 1->2->3->4->5->6->7->8->9->10 \*/

## struct Node\* head = NULL;

## int M=2, N=3;

## push(&head, 10);

## push(&head, 9);

## push(&head, 8);

## push(&head, 7);

## push(&head, 6);

## push(&head, 5);

## push(&head, 4);

## push(&head, 3);

## push(&head, 2);

## push(&head, 1);

## 

## printf("M = %d, N = %d \nGiven Linked list is :\n", M, N);

## printList(head);

## 

## skipMdeleteN(head, M, N);

## 

## printf("\nLinked list after deletion is :\n");

## printList(head);

## 

## return 0;

## }