**NAME: ASHWINI SHIVAJI RATHOD**

**CLASS:SE**

**DIV:3**

**Experiment no 7**

**Aim:** Implementation of Circular Linked List ADT

**Objective:** Circular Linked List can be used to manage the Computing Resources of the computer. Data Structure such as stacks and queue are implemented with the help of circular linked list

**Theory :**

1)The code defines a structure struct Node to represent a node in the circular linked list. Each node contains an integer value (data) and a pointer to the next node (next).

2)The push function is used to insert a new node at the beginning of the circular linked list. It takes a reference to the head of the list and an integer data as parameters. It creates a new node, updates its data and next pointer, and adjusts the pointers to maintain the circular structure.

3)The printList function is used to print the elements of the circular linked list. It traverses the list starting from the head and continues until it reaches the head again.

4)The deleteNode function is used to delete a node with a given key from the circular linked list. It takes a reference to the head of the list and the key to be deleted as parameters. It handles various cases, including deleting the head node, deleting a node in the middle, and deleting the last node in the list.

5)In the main function, a circular linked list is created by inserting nodes with values 2, 5, 7, 8, and 10 using the push function. Then, it prints the list before and after deleting a node with a key of 7 using the deleteNode function.

6)The code outputs the list before deletion and the list after deletion to demonstrate the effect of the deletion operation.

**Algorithm:**

1)Define a structure struct Node to represent nodes in the circular linked list. Each node contains an integer data and a pointer next to the next node in the list.

2)Implement a push function to insert a new node with the given data at the beginning of the circular linked list:

-Allocate memory for a new node (ptr1).

-Set the data and next pointer of the new node.

-If the linked list is not empty, find the last node and update its next pointer to point to the new

node.

-Update the head pointer to the new node.

3)Implement a printList function to print the elements of the circular linked list:

-Traverse the list from the head to the last node using a do-while loop.

-Print the data of each node until the loop reaches the head again.

4)Implement a deleteNode function to delete a node with a specified key (value) from the circular linked list:

-Handle the case when the list is empty.

-Handle the case when there is only one node in the list.

-Handle the case when the node to be deleted is the head node.

-Traverse the list to find the node with the specified key.

-Update the next pointer of the previous node to skip the node to be deleted.

-Free the memory of the deleted node.

5)In the main function:

-Initialize an empty circular linked list (head is initially set to NULL).

-Push elements (2, 5, 7, 8, 10) into the list using the push function.

-Print the list before deletion.

-Delete a node with the value 7 using the deleteNode function.

-Print the list after deletion.

**Code :**

#include <stdio.h>

#include <stdlib.h>

// Structure for a node

struct Node {

int data;

struct Node\* next;

};

// Function to insert a node at the

// beginning of a Circular linked list

void push(struct Node\*\* head\_ref, int data)

{

// Create a new node and make head

// as next of it.

struct Node\* ptr1 = (struct Node\*)malloc(sizeof(struct Node));

ptr1->data = data;

ptr1->next = \*head\_ref;

// If linked list is not NULL then

// set the next of last node

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

// Find the node before head and

// update next of it.

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

while (temp->next != \*head\_ref)

temp = temp->next;

temp->next = ptr1;

}

else

// For the first node

ptr1->next = ptr1;

\*head\_ref = ptr1;

}

// Function to print nodes in a given

// circular linked list

void printList(struct Node\* head)

{

struct Node\* temp = head;

if (head != NULL) {

do {

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

temp = temp->next;

} while (temp != head);

}

printf("\n");

}

// Function to delete a given node

// from the list

void deleteNode(struct Node\*\* head, int key)

{

// If linked list is empty

if (\*head == NULL)

return;

// If the list contains only a

// single node

if ((\*head)->data == key && (\*head)->next == \*head) {

free(\*head);

\*head = NULL;

return;

}

struct Node \*last = \*head, \*d;

// If head is to be deleted

if ((\*head)->data == key) {

// Find the last node of the list

while (last->next != \*head)

last = last->next;

// Point last node to the next of

// head i.e. the second node

// of the list

last->next = (\*head)->next;

free(\*head);

\*head = last->next;

return;

}

// Either the node to be deleted is

// not found or the end of list

// is not reached

while (last->next != \*head && last->next->data != key) {

last = last->next;

}

// If node to be deleted was found

if (last->next->data == key) {

d = last->next;

last->next = d->next;

free(d);

}

else

printf("Given node is not found in the list!!!\n");

}

// Driver code

int main()

{

// Initialize lists as empty

struct Node\* head = NULL;

// Created linked list will be

// 2->5->7->8->10

push(&head, 2);

push(&head, 5);

push(&head, 7);

push(&head, 8);

push(&head, 10);

printf("List Before Deletion: ");

printList(head);

deleteNode(&head, 7);

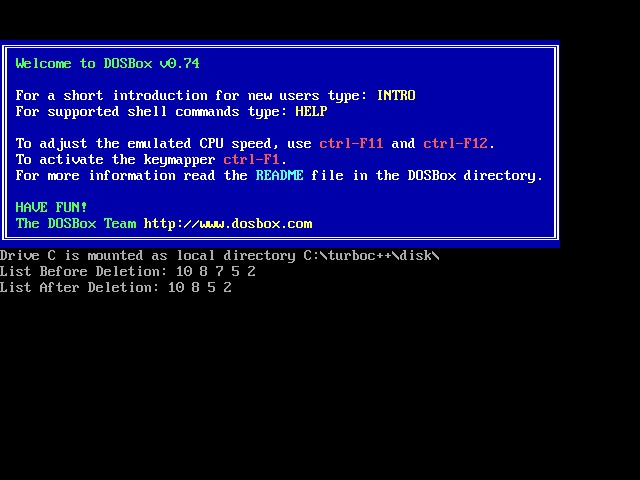
printf("List After Deletion: ");

printList(head);

return 0;

}

**Output :**



**Conclusion :**

This C code illustrates the creation and manipulation of a Circular Linked List. It includes functions to insert nodes at the beginning of the list and delete nodes by a specified key. The code creates a circular linked list, prints it before and after deleting a node, demonstrating the basic operations on such lists.