**NAME: UDIT BHURKE CLASS: SE**

**DIV: 03 ROLLNO: 03**

**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:**

The circular linked list is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which forms a circle. There is no NULL at the end.

**Algorithm:**

Insertion at the beginning:

* Step 1: NOW IF POINTER = NULL (GO TO STEP 11)
* Step 2: ASSIGN NEW NODE = POINTER
* Step 3: ASSIGN POINTER = POINTER → NEXT
* Step 4: ASSIGN NEW NODE → DATA = VALUE
* Step 5: ASSIGN TEMP NODE = HEAD
* Step 6: REPEAT STEP 8 WHILE TEMP → NEXT != HEAD (END OF WHILE LOOP)
* Step 7: ASSIGN TEMP NODE = TEMP → NEXT
* Step 8: ASSIGN NEW NODE → NEXT = HEAD
* Step 9: ASSIGN TEMP NODE → NEXT = NEW NODE
* Step 10: ASSIGN HEAD = NEW NODE
* Step 11: EXIT

Deletion at beginning:

* **Step 1:** IF HEAD = NULL
* Write UNDERFLOW  
   Go to Step 8  
   [END OF IF]**Step 2:** SET PTR = HEAD
* **Step 3:** Repeat Step 4 while PTR → NEXT != HEAD
* **Step 4:** SET PTR = PTR → next
* [END OF LOOP]**Step 5:** SET PTR → NEXT = HEAD → NEXT
* **Step 6:** FREE HEAD
* **Step 7:** SET HEAD = PTR → NEXT
* **Step 8:** EXIT

**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)

{

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

// 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;

}

// 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);

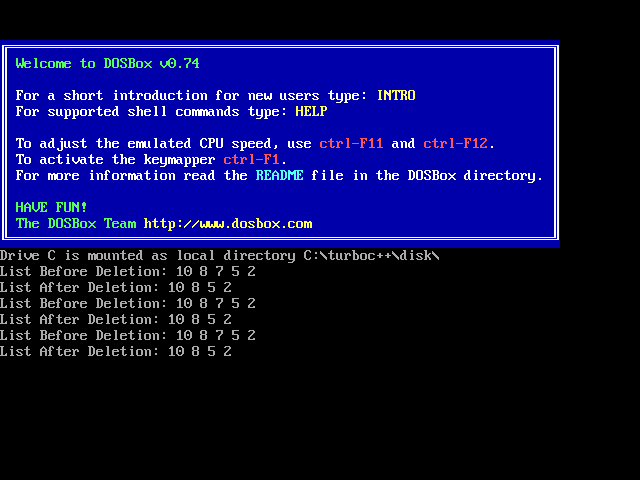
printf("List After Deletion: ");

printList(head);

return 0;

}

**Output:**



**Conclusion:** Thus circular linked list is implemented.