Circular Linked List Program in C

Circular Linked List is a variation of Linked list in which the first element points to the last element and the last element points to the first element. Both Singly Linked List and Doubly Linked List can be made into a circular linked list.

Implementation in C

Live Demo

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;

   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;

bool isEmpty() {
   return head == NULL;
}

int length() {
   int length = 0;

   //if list is empty
   if(head == NULL) {
      return 0;
   }

   current = head->next;

   while(current != head) {
```

```
      length++;
      current = current->next;
   }

   return length;
}

//insert link at the first location
void insertFirst(int key, int data) {

   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct node));
   link->key = key;
   link->data = data;

   if (isEmpty()) {
      head = link;
      head->next = head;
   } else {
      //point it to old first node
      link->next = head;

      //point first to new first node
      head = link;
   }
}

//delete first item
struct node * deleteFirst() {

   //save reference to first link
   struct node *tempLink = head;

   if(head->next == head) {
      head = NULL;
      return tempLink;
   }

   //mark next to first link as first
   head = head->next;

   //return the deleted link
   return tempLink;
}
```

```
//display the list
void printList() {

   struct node *ptr = head;
   printf("\n[ ");

   //start from the beginning
   if(head != NULL) {

      while(ptr->next != ptr) {
         printf("(%d,%d) ",ptr->key,ptr->data);
         ptr = ptr->next;
      }
   }

   printf(" ]");
}

void main() {
   insertFirst(1,10);
   insertFirst(2,20);
   insertFirst(3,30);
   insertFirst(4,1);
   insertFirst(5,40);
   insertFirst(6,56);

   printf("Original List: ");

   //print list
   printList();

   while(!isEmpty()) {
      struct node *temp = deleteFirst();
      printf("\nDeleted value:");
      printf("(%d,%d) ",temp->key,temp->data);
   }

   printf("\nList after deleting all items: ");
   printList();
}
```

If we compile and run the above program, it will produce the following result −

Output

```
Original List:
[ (6,56) (5,40) (4,1) (3,30) (2,20) ]
Deleted value:(6,56)
Deleted value:(5,40)
Deleted value:(4,1)
Deleted value:(3,30)
Deleted value:(2,20)
Deleted value:(1,10)
List after deleting all items:
[ ]
```