```
// Created by Ritwik Chandra Pandey on 13/02/21
//183215
//CIRCULAR LINKED LIST


#include <stdio.h>
#include <stdlib.h>


struct node {
   int data;
   struct node *next;
};

typedef struct node * NODE;
NODE first = NULL;

NODE createNodeInCLL(){
   NODE temp;
   temp = (NODE) malloc(sizeof(struct node));
   temp->next = NULL;


   return temp;}

NODE addNodesInCLL(NODE first, int x) {
   NODE temp, lastNode = first;
   temp = createNodeInCLL();
   temp -> data = x;
   if (first == NULL) {
      first = temp;
   } else {
      while (lastNode -> next != first) {
         lastNode = lastNode -> next;
      }
      lastNode -> next = temp;
   }
   temp -> next = first;
   return first;
```

```
}


int countInCLL(NODE first) {
   NODE temp = first;
   int sum = 0;
   if (first == NULL) {
      return sum;
   } else {
      do {
         sum++;
         temp = temp -> next;
      } while (temp != first);
      return sum;
   }
}

NODE insertAtBeginInCLL(NODE first, int x) {
   NODE temp, lastNode = first;
   temp = createNodeInCLL();
   temp -> data = x;
   if (first == NULL) {
      first = temp;
      temp -> next = first;
   } else {
      while (lastNode -> next != first) {
         lastNode = lastNode -> next;
      }
      temp -> next = first;
      first = temp;
      lastNode -> next = first;
   }
   return first;
}
NODE insertAtEndInCLL(NODE first, int x) {
   NODE temp, lastNode = first;
   temp = createNodeInCLL();
   temp -> data = x;
   if (first == NULL) {
      first = temp;
```

```
    } else {
        while (lastNode -> next != first) {
            lastNode = lastNode -> next;
        }
        lastNode -> next = temp;
    }
    temp -> next = first;
    return first;
}
void traverseListInCLL(NODE first) {
    NODE temp = first;
    do {
        printf("%d --> ", temp -> data);
        temp = temp -> next;
    } while (temp != first);
    printf("END");
    printf("\n");
}


int searchPosOfEleInCLL(NODE first, int key) {
                            NODE currentNode = first, q = first;
                            int count = 0;
                            if (currentNode == NULL) {
                                return count;
                            } else {
                                do {
                                    count++;
                                    q = currentNode;
                                    if (currentNode -> next == first && currentNode -> data != key) {
                                        return 0;
                                    }
                                    currentNode = currentNode -> next;
                                } while (q -> next != first && q -> data != key);
                                return count;
                            }
                        }
NODE deleteAtBeginInCLL(NODE first) {
                            NODE prev = first, lastNode = first;
                            if (prev -> next == first) {
```

```
                first = NULL;
            } else {
                while (lastNode -> next != first) {
                    lastNode = lastNode -> next;
                }
                first = prev -> next;
                lastNode -> next = first;
            }
            printf("The deleted item from Circular Linked List : %d\n", prev -> data);
            free(prev);
            return first;
        }

NODE deleteAtEndInCLL(NODE first) {
            NODE prev, lastNode = first;
            if (lastNode -> next == first) {
                first = NULL;
            } else {
                while (lastNode -> next != first) {
                    prev = lastNode;
                    lastNode = lastNode -> next;
                }
                prev -> next = first;
            }
            printf("The deleted item from Circular Linked List : %d\n", lastNode -> data);
            free(lastNode);
            return first;
        }

NODE deleteAtPositionInCLL(NODE first, int pos) {
    NODE prev = first, lastNode = first;
    int i;
    if (pos == 1) {
        if (prev -> next == first) {
            first = NULL;
        } else {
            while (lastNode -> next != first) {
                lastNode = lastNode -> next;
            }
            first = prev -> next;
            lastNode -> next = first;
        }
    } else {
        for (i = 1; i < pos; i++) {
            if (prev -> next == first) {
                printf("No such position in Circular Linked List. So deletion is not possible\”);
                return first;
            }
            lastNode = prev;
            prev = prev -> next;
        }
        lastNode -> next = prev -> next;
    }
    printf("The deleted item from Circular Linked List : %d\n”, prev -> data);
    free(prev);
    return first;
}

NODE insertAtPositionInCLL(NODE first, int pos, int x) {
    NODE temp, lastNode = first;
    int i ;
    for (i = 1; i < (pos - 1); i++) {
        if (lastNode -> next == first) {
            printf("No such position in Circular Linked List. So insertion is not possible\n");
            return first;
        }
        lastNode = lastNode -> next;
    }
    temp = createNodeInCLL();
    temp -> data = x;
    if (pos == 1) {
        if (first == NULL) {
            first = temp;
            temp -> next = first;
        } else {
            while (lastNode -> next != first) {
                lastNode = lastNode -> next;
            }
            temp -> next = first;
            first = temp;
            lastNode -> next = first;
        }
    } else {
        temp -> next = lastNode -> next;
        lastNode -> next = temp;
    }
    return first;
}
NODE deleteList(NODE head_ref)
{
    NODE current = head_ref;
    NODE next;
    do{


        next = current->next;
        free(current);
        current = next;
    }while(current!=head_ref);
    head_ref = NULL;
    return head_ref; }

                int main(){

                    int selection=0,x,pos;
                    printf("\t\tCIRCULAR LINKED LIST\n\n");

                    do{



                        printf("\t1.ADD NODES\n\t2.COUNT\n\t3.INSERT AT BEGIN\n\t4.INSERT AT END\n\t5.INSERT AT
POSITION\n\t6.TRAVERSE LIST\n\t7.SEARCH\n\t8.DELETE AT BEGIN\n\t9.DELETE AT END\n\t10.DELETE AT POSITION\n\t11.DELETE
LIST\n\t12.EXIT\n");

                        printf("\t\n Please enter your choice\n");
                        scanf("%d",&selection);


                     switch(selection)
```

```
{

      case 1:

              printf("Enter an element : Put -1 to Stop ");
              scanf("%d", &x);
     while(x!=-1){
              first = addNodesInCLL(first,x);
              printf("Enter an element : ");
       scanf("%d",&x);}
              break;

      case 2:
          printf("The number of nodes in a Circular Linked List are : %d\n", countInCLL(first));
              break;
      case 3:

          printf("Enter an element : ");
          scanf("%d", &x);
              first = insertAtBeginInCLL(first, x);
          break;
      case 4:

          printf("Enter an element : ");
          scanf("%d", &x);
              first = insertAtEndInCLL(first, x);
          break;
      case 5:

          printf("Enter a position : ");
          scanf("%d", &pos);
          printf("Enter an element : ");
          scanf("%d", &x);
          if (pos <= 0 || (pos > 1 && first == NULL)) {
              printf("No such position in Circular "
                     "Linked List. So insertion is not possible\n");
          } else {
              first = insertAtPositionInCLL(first, pos, x);
          }
               break;
```

```
case 6:
  if(first == NULL){
    printf("Circular Linked List is empty\n");
  }else{
    printf("The elements in Circular Linked List are: ");
    traverseListInCLL(first);
  }
  break;

case 7:

  printf("Enter a search element : ");
  scanf("%d", &x);
  pos = searchPosOfEleInCLL(first, x);
  if (pos == 0) {
    printf("The given element %d is not found in the given Circular Linked List.", x);
  } else {
    printf("The given element %d is found at position %d", x, pos);
  }
    break;
case 8:
  if(first == NULL){
    printf("Linked List is empty. So deletion is not possible");
  }else{
    first = deleteAtBeginInCLL(first);
  }
  break;
case 9:
  if (first == NULL) {
    printf("Linked List is empty. "
        "So deletion is not possible");
  } else {
    first = deleteAtEndInCLL(first);
  }
  break;
case 10:

  if (first == NULL) {
```

```
                    printf("Linked List is empty. "
                           "So deletion is not possible");
                } else {
                    printf("Enter a position : ");
                    scanf("%d", &pos);
                    if (pos <= 0) {
                        printf("No such position in Circular "
                               "Linked List. So deletion is not possible");
                    } else {
                        first = deleteAtPositionInCLL(first, pos);
                    }
                }
                  break;
        case 11:
            first=deleteList(first);
            break;

        case 12:
            break;
              default:
                  printf("\t\n\nYou have not entered the right choice\n\n");
    }}
          while(selection!=12);

}
```