Se dan las estructuras siguientes:

```
struct Tnodo {
    int dato;
    struct Tnodo * siguiente;
} ;

struct TlistaNodos {
   struct Tnodo * inicio;
} ;
```

para definir una lista de nodos circular simplemente enlazados. Implementar las funciones descritas en la sección de prototipos, probando su funcionamiento en un programa.

SOLUCIÓN:

```
//defines
#include <stdio.h>
#include <stdlib.h>


//estructuras
typedef struct Nodo {
    int dato;
    struct Nodo * siguiente;
} NODO;

typedef struct LstNodo {
    NODO * head;
    NODO * tail;
} LST_NODO;


//prototipos
void iniciarLista(LST_NODO *);
void insertarListaCircular(int, LST_NODO *);
void recorrerListaCircular(LST_NODO);
void clearListaCircular(LST_NODO *);


//main
int main(void)
{
    LST_NODO lst;
    iniciarLista(&lst);
    insertarListaCircular(5, &lst);
    insertarListaCircular(5, &lst);
    clearListaCircular(&lst);
    recorrerListaCircular(lst);
    return 0;
}

//implementacion de las funciones
void iniciarLista(LST_NODO * lst)
{
    lst->head = NULL;
    lst->tail = NULL;
}

void insertarListaCircular(int value, LST_NODO * lst)
{
    if (lst->tail)
    {
```

```
        lst->tail->siguiente = (NODO *) malloc(sizeof(LST_NODO));
        lst->tail->siguiente->siguiente = lst->head;
        lst->tail->siguiente->dato = value;
        lst->tail = lst->tail->siguiente;
    }
    else
    {
        lst->head = (NODO *) malloc(sizeof(LST_NODO));
        lst->head->siguiente = lst->head;
        lst->head->dato = value;
        lst->tail = lst->head;
    }
}

void recorrerListaCircular(LST_NODO lst)
{
    NODO * curr = lst.head;
    if (curr)
    {
        printf("%d ", curr->dato);
        while(curr != lst.tail)
        {
            curr = curr->siguiente;
            printf("\n%d", curr->dato);
        }
    }
}

void clearListaCircular(LST_NODO * lst)
{
    NODO * curr = lst->head;
    if (curr)
    {
        lst->head = lst->head->siguiente;
        free(curr);
        while (curr != lst->tail)
        {
            curr = lst->head;
            lst->head = lst->head->siguiente;
            free(curr);
        }
    }
    free(lst->tail);
    lst->head = NULL;
    lst->tail = NULL;
}
```