C function of Circular linked list

```
#include <stdio.h>
#include <conio.h>
#include<malloc.h>
struct node
  {
    int data;
    struct node *next;
  };
 struct node * start = NULL;
 struct node * create_cll (struct node *);
 struct node * display (struct node *);
 struct node * insert_beg (struct node *);
 struct node * insert_end (struct node *);
 struct node * delete_beg(struct node *);
 struct node * delete_end (struct node *);
 struct node * delete_after (struct node *);
 struct node * delete_list (struct node *);
 int main ()
  {
   int option;
   clrscr();
   do
   {
    printf("\n 1: Create a list");
    printf("\n 2: Display a list");
    printf ("\n 3: Add node at Begin");
    printf("\n 4: "  "  "  " End);
    printf("\n 5: Delete node from Begin");
    printf("\n 6. Delete a node from End");
    printf("\n 7: Delete a node after a given node");
    printf("\n 8: Delete the entire list");
```

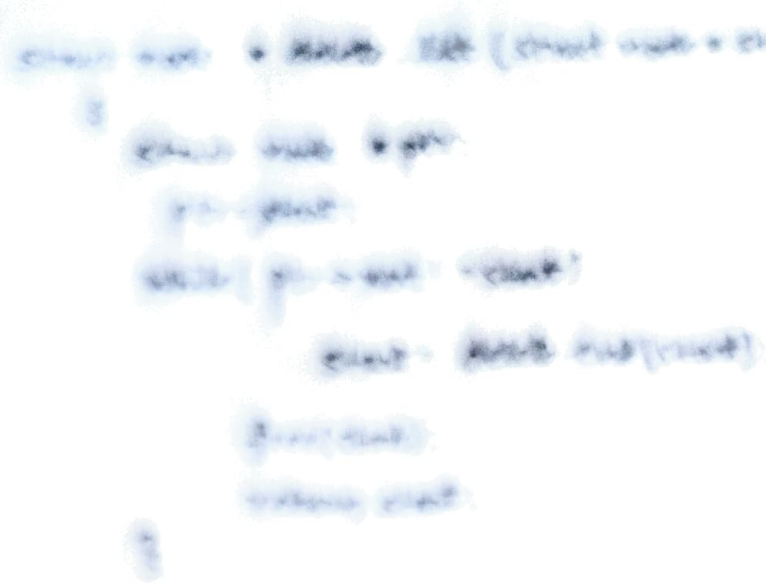
```
printf("\n 9: Exit");
printf("\n\n Enter your option: ");
scanf ("%d", &option);
switch (option)
  {
    case 1: start = creak_cll (start);
            printf("\n Circular Linked list created");
            break;

    case 2: start = display (start);
            break;

    case 3; Start = insert_beg(start);
            break;

    case 4: start = insert_end (start);
            break;

    case 5: start = delete_beg (start);
            break;

    case 6: start = delete_end (start);
            break;

    case 7: start = delete_after(start);
            break;

    case 8: start = delete_list (start);
            printf("\n Circular linked list deleted");
            break;

  }

} while(        )

   getch( );
   return 0;
}
```

```
struct node *create_cll(struct node *ptr)
{
  struct node *new_node, *ptr;
  int num;
  printf("\n Enter -1 to end");
  printf("\n Enter the data");
  scanf("%d",&num);
  while(num!=-1)
    {
      new_node=(struct node *)malloc(sizeof(struct node));
      new_node->data=num;
      if(start==NULL)
        {
          new_node->next=new_node;
          start=new_node;
        }
      else
        {
          ptr=start;
          while(ptr->next!=start)
            {
              ptr=ptr->next;
            }
          ptr->next=newnode;
          new_node->next=start;
        }
        printf("\n Enter the data:");
        scanf("%d",&num);
    }
  return start;
}
```

```
struct node * display (struct node * start)
  {
    struct node * ptr;
     ptr = start;
     while (ptr -> next != start)
        {
         printf ("\t %d", ptr -> data);
           ptr = ptr -> next;
        }
       return start;
  }

Struct node * insert_beg (struct node * start)
  {
    struct node * new_node, *ptr;
    int num;
    printf ("\n Enter the data: ");
    scanf ("%d", &num);
     new_node = (struct node*) malloc (sizeof (struct node));
       new_node -> data = num;
       ptr = start;
       while( ptr -> next != start)
            {ptr = ptr -> next;
              }
          ptr -> next = new_node;
           new_node -> next = start;
             start = new_node;
               return start;
          }
```

```
struct node * insert_end( struct node * start)
  {
   struct node *ptr, *new_node;
   int num;
   printf ("\n Enter the data:");
   scanf ("%d", &num);
   new_node = (struct node *) malloc (sizeof (struct node));
   new_node -> data = num;
    ptr = start;
    while( ptr -> next != start)
          {
           ptr = ptr -> next;
          }
        ptr -> next = new_node;
        new_node -> next = start;
        return start;
      }

struct node * delete_beg (struct node * start)
    {
      struct node * ptr;
        ptr = start;
      while (ptr -> next != start)
            {
              ptr = ptr -> next;
            }
        ptr -> next = start -> next;
          free (start)
           start = ptr -> next
           free(ptr);
          return start;
        }
```

```
Struct node * delete_end( struct node *start)
    {
      struct node *ptr, *preptr;
       ptr = start;
      while (ptr -> next != start)
          {
            preptr = ptr ;
            ptr = ptr -> next;
          }
        preptr -> next = ptr -> next;
        free (ptr);
         return start;
      }

Struct node * delete_after (struct node * start)
    {
      struct node *ptr, *preptr;
      int val;
      printf ("\n Enter the value after which you want to
                delete node");
       scanf ("%d", &val);
        ptr = start;
         preptr = ptr;
         while (preptr -> data != val)
            {
              preptr = ptr;
              ptr = ptr -> next;
            }
          preptr -> next = ptr -> next;
          if (ptr == start)
                {
                  start = preptr -> next;
                free (ptr);  return start;
```



```
struct node * delete_list (struct node * start)
{
    struct node *ptr;
    ptr = start;
    while ( ptr -> next != start)
            start = delete_end (start);
        free(start);
        return start;
}
```