**Assignment 2**

1.Write a Program to insert a node at the beginning of a circular Linked List.

Ans.

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

struct node {
   int num;
   struct node * nextptr;
}*stnode;

void ClListcreation(int n);
void ClLinsertNodeAtBeginning(int num);
void displayClList(int a);

int main()
{
   int n,num1,a;
   stnode = NULL;
        printf("\n\n Circular Linked List : Insert a node at the beginning of a circular linked list
:\n");
        printf("--------------------------------------------------------------------------------\n");
   printf(" Input the number of nodes : ");
   scanf("%d", &n);
   ClListcreation(n);
   a=1;
   displayClList(a);
        printf(" Input data to be inserted at the beginning : ");
   scanf("%d", &num1);
   ClLinsertNodeAtBeginning(num1);
   a=2;
   displayClList(a);
   return 0;
}

void ClListcreation(int n)
{
   int i, num;
   struct node *preptr, *newnode;

   if(n >= 1)
   {
      stnode = (struct node *)malloc(sizeof(struct node));

      printf(" Input data for node 1 : ");
```

```
        scanf("%d", &num);
        stnode->num = num;
        stnode->nextptr = NULL;
        preptr = stnode;
        for(i=2; i<=n; i++)
        {
            newnode = (struct node *)malloc(sizeof(struct node));
            printf(" Input data for node %d : ", i);
            scanf("%d", &num);
            newnode->num = num;
            newnode->nextptr = NULL;    // next address of new node set as NULL
            preptr->nextptr = newnode;  // previous node is linking with new node
            preptr = newnode;           // previous node is advanced
        }
        preptr->nextptr = stnode;                   //last node is linking with first node
    }
}

void ClLinsertNodeAtBeginning(int num)
{
    struct node *newnode, *curNode;
    if(stnode == NULL)
    {
        printf(" No data found in the List yet.");
    }
    else
    {
        newnode = (struct node *)malloc(sizeof(struct node));
        newnode->num = num;
        newnode->nextptr = stnode;
        curNode = stnode;
        while(curNode->nextptr != stnode)
        {
            curNode = curNode->nextptr;
        }
        curNode->nextptr = newnode;
        stnode = newnode;
    }
}
void displayClList(int m)
{
    struct node *tmp;
    int n = 1;
```

```
    if(stnode == NULL)
    {
        printf(" No data found in the List yet.");
    }
    else
    {
        tmp = stnode;
        if (m==1)
        {
        printf("\n Data entered in the list are :\n");
        }
        else
        {
         printf("\n After insertion the new list are :\n");
        }
        do {
            printf(" Data %d = %d\n", n, tmp->num);
            tmp = tmp->nextptr;
            n++;
        }while(tmp != stnode);
    }
}
```

## 2.Write a Program to insert a node at the end of a circular Linked List.

Ans.

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

struct node {
    int num;
    struct node * nextptr;
}*stnode;

struct node *tail,*p,*q,*store;

void ClListcreation(int n);
void ClLinsertNodeAtEnd(int num);
void displayClList(int a);

int main()
{
    int n,num1,a,insPlc;
    stnode = NULL;
            printf("\n\n Circular Linked List : Insert a node at the end of a circular linked list :\n");
            printf("--------------------------------------------------------------------------------\n");
```

```
    printf(" Input the number of nodes : ");
    scanf("%d", &n);
    ClListcreation(n);
    a=1;
    displayClList(a);
    printf(" Input the data to be inserted : ");
    scanf("%d", &num1);
    ClLinsertNodeAtEnd(num1);
    a=2;
    displayClList(a);
    return 0;
}

void ClListcreation(int n)
{
    int i, num;
    struct node *preptr, *newnode;
    if(n >= 1)
    {
        stnode = (struct node *)malloc(sizeof(struct node));
        printf(" Input data for node 1 : ");
        scanf("%d", &num);
        stnode->num = num;
        stnode->nextptr = NULL;
        preptr = stnode;
        for(i=2; i<=n; i++)
        {
            newnode = (struct node *)malloc(sizeof(struct node));
            printf(" Input data for node %d : ", i);
            scanf("%d", &num);
            newnode->num = num;
            newnode->nextptr = NULL;    // next address of new node set as NULL
            preptr->nextptr = newnode;  // previous node is linking with new node
            preptr = newnode;           // previous node is advanced
        }
        preptr->nextptr = stnode;                   //last node is linking with first node
    }
}

void ClLinsertNodeAtEnd(int num1)
{
                    int a;
                    a=num1;
                    struct node *temp=(struct node*)malloc(sizeof(struct node));
```

```
                temp->num=a;
                p=stnode;
                while(p->nextptr!=stnode)
                {
                        p=p->nextptr;
                }
                p->nextptr=temp;
                temp->nextptr=stnode;
}

void displayClList(int m)
{
   struct node *tmp;
   int n = 1;
   if(stnode == NULL)
   {
      printf(" No data found in the List yet.");
   }
   else
   {
      tmp = stnode;
      if (m==1)
      {
      printf("\n Data entered in the list are :\n");
      }
      else
      {
       printf("\n After insertion the new list are :\n");
      }
      do {
         printf(" Data %d = %d\n", n, tmp->num);

         tmp = tmp->nextptr;
         n++;
      }while(tmp != stnode);
   }
}
```

3.Write a Program to search a node in a circular Linked List.

Ans.#include <stdio.h>

#include <stdlib.h>

struct node {

```
    int num;
    struct node * nextptr;
}*stnode,*ennode;


void ClListcreation(int n);
int FindElement(int FindElem, int n);
void displayClList();

int main()
{
    int n,m;
    int i,FindElem,FindPlc;
    stnode = NULL;
    ennode = NULL;
        printf("\n\n Circular Linked List : Search an element in a circular
linked list :\n");
        printf("--------------------------------------------------------------------\n");

    printf(" Input the number of nodes : ");
    scanf("%d", &n);
    m=n;

    ClListcreation(n);
    displayClList();
    printf(" Input the element you want to find : ");
    scanf("%d", &FindElem);

    FindPlc=FindElement(FindElem,m);
        if(FindPlc<n)
                printf(" Element found at node %d \n\n",FindPlc);
        else
                printf(" This element does not exists in linked list.\n\n");
```

```
    return 0;
}

void ClListcreation(int n)
{
    int i, num;
    struct node *preptr, *newnode;

    if(n >= 1)
    {
        stnode = (struct node *)malloc(sizeof(struct node));

        printf(" Input data for node 1 : ");
        scanf("%d", &num);
        stnode->num = num;
        stnode->nextptr = NULL;
        preptr = stnode;
        for(i=2; i<=n; i++)
        {
            newnode = (struct node *)malloc(sizeof(struct node));
            printf(" Input data for node %d : ", i);
            scanf("%d", &num);
            newnode->num = num;
            newnode->nextptr = NULL;   // next address of new node set as NULL
            preptr->nextptr = newnode;  // previous node is linking with new
node
            preptr = newnode;               // previous node is advanced
        }
        preptr->nextptr = stnode;                   //last node is linking with first
node
    }
}
```

```
int FindElement(int FindElem, int a)
{
        int ctr=1;
        ennode=stnode;
        while(ennode->nextptr!=NULL)
        {
                if(ennode->num==FindElem)
                        break;
                else
                        ctr++;
                        ennode=ennode->nextptr;
                        if (ctr==a+1)
                        break;
        }
        return ctr;
}

void displayClList()
{
   struct node *tmp;
   int n = 1;

   if(stnode == NULL)
   {
      printf(" No data found in the List yet.");
   }
   else
   {
      tmp = stnode;
      printf("\n\n Data entered in the list are :\n");

      do {
```

```
        printf(" Data %d = %d\n", n, tmp->num);

        tmp = tmp->nextptr;
        n++;
      }while(tmp != stnode);
   }
}
```

4.Write a Program to delete a node in a circular Linked List.
Ans.Python program to delete a given key from
# linked list.

```
# Node of a doubly linked list

class Node:

    def __init__(self, next = None, data = None):

        self.next = next

        self.data = data

# Function to insert a node at the beginning of
# a Circular linked list

def push(head_ref, data):


    # Create a new node and make head as next

    # of it.

    ptr1 = Node()
```

```
ptr1.data = data

ptr1.next = head_ref


# If linked list is not None then set the

# next of last node

if (head_ref != None) :



    # Find the node before head and update

    # next of it.

    temp = head_ref

    while (temp.next != head_ref):

        temp = temp.next

    temp.next = ptr1



else:

    ptr1.next = ptr1 # For the first node
```

```
    head_ref = ptr1

    return head_ref

# Function to print nodes in a given
# circular linked list

def printList( head):


    temp = head

    if (head != None) :

        while(True) :

            print( temp.data, end = " ")

            temp = temp.next

            if (temp == head):

                break

    print()

# Function to delete a given node from the list

def deleteNode( head, key) :


    # If linked list is empty
```

```
if (head == None):
    return

# If the list contains only a single node
if((head).data == key and (head).next == head):

    head = None

last = head
d = None

# 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
    head = last.next

# Either the node to be deleted is not found
# or the end of list is not reached
while(last.next != head and 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

else:
```

```
        print("no such keyfound")


    return head

# Driver code

# Initialize lists as empty

head = None

# Created linked list will be 2.5.7.8.10

head = push(head, 2)

head = push(head, 5)

head = push(head, 7)

head = push(head, 8)

head = push(head, 10)


print("List Before Deletion: ")
printList(head)


head = deleteNode(head, 7)
```

print( "List After Deletion: ")
printList(head)

5.Write a program that should take the key number as input and delete that particular node.
Ans.
Linked List | Set 3 (Deleting a node)
We have discussed Linked List Introduction and Linked List Insertion in previous posts on a singly linked list.
Let us formulate the problem statement to understand the deletion process. Given a ‘key’, delete the first occurrence of this key in the linked list.

Iterative Method:
To delete a node from the linked list, we need to do the following steps.
1) Find the previous node of the node to be deleted.
2) Change the next of the previous node.
3) Free memory for the node to be deleted.

linkedlist_deletion

Recommended Practice
Delete a Node in Single Linked List
Try It!
Since every node of the linked list is dynamically allocated using malloc() in C, we need to call free() for freeing memory allocated for the node to be deleted.

```
// A complete working C program
// to demonstrate deletion in
// singly linked list
#include <stdio.h>
```

```
#include <stdlib.h>

// A linked list node

struct Node {

    int data;

    struct Node* next;
};

/* Given a reference (pointer to pointer) to the head of a

   list and an int, inserts a new node on the front of the

   list. */

void push(struct Node** head_ref, int new_data)
{

    struct Node* new_node

        = (struct Node*)malloc(sizeof(struct Node));

    new_node->data = new_data;

    new_node->next = (*head_ref);

    (*head_ref) = new_node;
}

/* Given a reference (pointer to pointer) to the head of a
```

```
   list and a key, deletes the first occurrence of key in
   linked list */
void deleteNode(struct Node** head_ref, int key)
{
    // Store head node
    struct Node *temp = *head_ref, *prev;

    // If head node itself holds the key to be deleted
    if (temp != NULL && temp->data == key) {
        *head_ref = temp->next; // Changed head
        free(temp); // free old head
        return;
    }

    // Search for the key to be deleted, keep track of the
    // previous node as we need to change 'prev->next'
    while (temp != NULL && temp->data != key) {
        prev = temp;
```

```
        temp = temp->next;
    }

    // If key was not present in linked list
    if (temp == NULL)
        return;

    // Unlink the node from linked list
    prev->next = temp->next;

    free(temp); // Free memory
}

// This function prints contents of linked list starting
// from the given node
void printList(struct Node* node)
{
    while (node != NULL) {
        printf(" %d ", node->data);
        node = node->next;
    }
```

```
}

// Driver code

int main()
{

    /* Start with the empty list */

    struct Node* head = NULL;


    push(&head, 7);

    push(&head, 1);

    push(&head, 3);

    push(&head, 2);


    puts("Created Linked List: ");

    printList(head);

    deleteNode(&head, 1);

    puts("\nLinked List after Deletion of 1: ");

    printList(head);

    return 0;
```