Outline
- Circular Linked Lists
  - Circular Linked Lists
  - Primitive Functions in Circular Lists
  - Application of the Circular Lists

Circular Linked Lists

In linear linked lists if a list is traversed (all the elements visited) an external pointer to the list must be preserved in order to be able to reference the list again.

Circular linked lists can be used to help the traverse the same list again and again if needed. A circular list is very similar to the linear list where in the circular list the pointer of the last node points not NULL but the first node.

---

**Linear Linked List**

```
list
  info
  info
  info
  info null
```

*a linear linked list*

---

**Circular Linked List without Header Node**

```
first node
  info
  info
  info
  last node
```

*a circular linked list without header node*

---

**Circular Linked List with a Header Node**

```
header node
  info
  info
  info
```

*a circular linked list with a header node*
In a circular linked list there are two methods to know if a node is the first node or not.
- Either a external pointer, list, points the first node or
- A header node is placed as the first node of the circular list.

The header node can be separated from the others by either having a sentinel value as the info part or having a dedicated flag variable to specify if the node is a header node or not.

**PRIMITIVE FUNCTIONS IN CIRCULAR LISTS**

The structure definition of the circular linked lists and the linear linked list is the same:

```c
struct node{
    int info;
    struct node *next;
};
typedef struct node nodeptr;
```

The delete after and insert after functions of the linear lists and the circular lists are almost the same.

The deleteafter function: **delafter()**

```c
void delafter(nodeptr *p, int *px)
{
    nodeptr *q;
    if((p == NULL) || (p == p->next)){ /*the empty list contains a single node and may be pointing itself*/
        printf("void deletion\n");
        exit(1);
    }
    q = p->next;
    *px = q->info; /*the data of the deleted node*/
    p->next = q->next;
    freenode(q);
}
```

The insertafter function: **insafter()**

```c
void insafter(nodeptr *p, int x)
{
    nodeptr *q;
    if(p == NULL){
        printf("void insertion\n");
        exit(1);
    }
    q = getnode();
    q->info = x; /*the data of the inserted node*/
    q->next = p->next;
    p->next = q;
}
```
Circular lists with a header node: The header node in a circular list can be specified by a sentinel value or a dedicated flag:

**Header Node with Sentinel:** Assume that info part contains positive integers. Therefore the info part of a header node can be -1. The following circular list is an example for a sentinel used to represent the header node:

```c
struct node{
    int info;
    struct node *next;
};
typedef struct node nodeptr;
```

![Diagram of a circular list with a header node using sentinel value]

**Header Node with Flag:** In this case a **flag** can be used to represent the header node. For example flag in the header node can be 1, where the flag is 0 for the other nodes.

```c
struct node{
    int flag;
    int info;
    struct node *next;
};
typedef struct node nodeptr;
```

![Diagram of a circular list with a header node using flag]

---

*3*
Ex: Consider a circular linked list with a header node, where each node contains the name, account number and the balance of a bank customer. The header node contains a sentinel account number to be -99.

(a) Write an appropriate node structure definition for the circular linked list.
(b) Write a function to display the full records of the customers with negative balance.

a) struct node{
    char Name[15];
    int AccNo;
    float Balance;
    struct node *next;
};
typedef struct node nodeptr;

b) Assume that the list pointer points the header with the sentinel account number -99.

void DispNegBalanca(nodeptr *plist)
{
    nodeptr *p;
    p=plist;
    if(p == NULL){
        printf("There is no list!\n");
        exit(1);
    }
    p=p->next;
    while(p->AccNo!=-99){
        if(p->Balance < 0.0)
            printf("The Customer Name:%s\nThe Account No:%d\nThe Balance:%.2f\n", p->Name, p->AccNo, p->Balance);
        p=p->next;
    }
}

Ex: Write a function that returns the average of the numbers in a circular list. Assume that the following node structure is used, where the flag variable is 1 for the header node and 0 for all the other nodes.

struct node{
    int flag;
    float info;
    struct node *next;
};
typedef struct node nodeptr;

float avList(nodeptr *plist)/*assume that plist points the header node*/{
    int count=0;
    float sum =0.0;
    nodeptr *p;
    p=plist;
    if((p == NULL)){
        printf("Empty list\n");
        exit(1);
    }
```c
    do{
        sum=sum + p->info;
        p =p->next;
        count++;
    }while(p->flag !=1);
    return sum/count;
    }
```