DSA
INTERNSHALA
Circular Linked List

Traversing a Circular Linked List

* The link of the last node contains address of first node instead of null thus making the list continuous or circular

```
[Algorithm to traverse a circular linked list]
   1. Write:data[start]
   2. Set temp:=link[start]
   3. Repeat steps 4,5 while temp!=start
   4. Write:data[temp]
   5. Set temp:=link[temp]
   6. Return
```

Inserting a new node in the beginning

* to do so we will also require the address of the last node as it is supposed to hold the address of the first node

```
[Algorithm to insert a new node in the beginning]
   1. If avail=NULL, then:
      Write:Overflow error
      Return
      [end of if]
   2. Set new:=avail, avail:=link[avail]
   3. Set data[new]:=val
   4. Set temp:=link[start]
   5. Repeat step 6 while temp != start     [to find the last node]
   6. Set temp:=link[temp]
   7. Set link[new]:=start
   8. Set start:=new
   9. Ste link[temp]:=start
   10.     Return
```

Inserting a new node before a given node

```
[Algorithm to insert a new node before a given node]
   1. If data[start]=item, then:
   2. Call insb  [call the algo to insert at begining]
   Exit
   [end of if]
3. If avail=NULL, then:
   Write: Overflow error
   Exit
   [end of if]
4. Set new:= avail, avail:=link[avail]
5. Set data[new]:=val
6. Set temp:=link[start], prev:=start
7. Repeat steps 8,9 while temp!=start
8. If data[temp]=item, then:
   Set link[new]:=temp
   Set link[prev]:=new
   Exit
   [end of if]
9. Set prev:= temp, temp:=link[temp]
10.     Write: No such node with value item exists
11.     Exit
```

Inserting a new node after a given node

[Algorithm to insert a new node after a given node]

```
1. If avail=NULL, then:
   Write: Overflow error
   Exit
   [end of if]
2. Set new:= avail, avail:=link[avail]
3. Set data[new]:=val
4. If data[start]=item, then
   Set link[new]:=link[start]
   Set link[start]=new
   Return
   [end of if]
5. Set temp:=link[start]
6. Repeat steps 7,8 while temp!=start
7. If data[temp]=item, then:
   Set link[new]:=link[temp]
   Set link[temp]:=new
   Exit
   [end of if]
8. Set temp:=link[temp]
9. Write: No such node with value item exists
10.     Exit
```

Deleting the first node of a Circular Linked List

[Algorithm to delete the first node in a circular linked list]

```
1. Set start1:=start     [preserving the address of first node]
2. Set temp:=start
3. Repeat step 4 while link[temp]!=start
4. Set temp:=link[temp]
5. Set start:=link[start]
6. Set link[temp]:=start
   [deleted node being inserted as the first node of available liked
   list]
7. Set link[start1]:=avail
8. Set avail:=start1
9. Return
```

Deleting a particular node in a Circular Linked List

[Algorithm to delete a particular node in a circular linked list]

```
1. If data[start]:=item, then:
   Call deleteFirst
   Exit
   [end of if]
2. Set temp:=link[start], prev:=start
3. Repeat step 4,5 while link[temp]!=start
4. If item=data[temp], then:
   Set link[prev]:=link[temp]
   Exit
   [end of if]
5. Set prev:=temp, temp:=link[temp]
6. Write: No such node with value item exists
7. Exit
```

Creation and traversal of a circular linked list

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

struct clist
{
      Int data;
      struct clist *link;
};
typedef struct clist node;

node * create(node *start)
{
      node *temp, *ptr;
      char ch;
      int num;
      do
      {
            printf("\n\t Enter the value of number: ");
            scanf("%d",&num);
            temp = (node*) malloc(sizeof(node));
            if(start==NULL)
            {
                  start=temp;
                  ptr=start;
            } else {
                  ptr->link = temp;
                  ptr = ptr->link;
            }
      temp->link =start;
      printf("\n\t Do you want to add more nodes (y/n):");
      fflush(stdin);
      scanf("%c",&ch);
      } while(ch=='y' || ch=='Y');
      return(start);
}

void display(node *start)
{
      node *temp;
      printf("\n\n Base address Number Link");
      printf("\n ==========================");
      printf("\n%10u %10d %10u",start,start->data,start->link);
      for(temp=start->link;temp!=start;temp=temp->link)
            printf("\n%10u %10d %10u",temp,temp->data,temp->link);
      getch();
      return;
}
```

Insertion of new node in the beginning

```
node *insetBeg(node *start)
{
	node *temp,*end,*ptr;
	int num;
	char ch;
	ptr = start;
	do
	{
		end = ptr;
		ptr = ptr->link;
	} while(ptr!=start);

	printf(“\n\n\t Enter the number to insert: ”);
	scanf(“%d”,&num);
	temp = (node*) malloc(sizeof(node));
	temp->data = num;
	temp->link=start;
	end->link =temp;
	start =temp;
	return(start);
}
```

Inserting a new node after a given node

```
node* search(node *start, int num)
{
	node *temp, *loc;
	loc = NULL;
	temp = start;
	do
	{
		if(num==temp->data)
		{
			loc = temp;
			break;
		}
		temp=temp->link;
	} while(temp!=start);
	return(loc);
}

void insetAfter(node *start)
{
	node *temp, *loc;
	int num, data;

	printf(“\n\n\t Enter the no. after which you want to insert:”);
```

```
        scanf("%d",&data);

        loc=search(start,data);
        if(loc==NULL)
                printf("\n\t The number is not present in the Linked
        list");
        else
        {
                printf("\n\n\t Enter the number to be inserted:");
                scanf("%d",&num);
                temp = (node*) malloc(sizeof(node));
                temp->data = num;
                temp->link = loc->link;
                loc->link = temp;
        }
}
```

Deleting a particular node

```
node* del(node *start)
{
        node *temp,*ptr,*end,*loc,*i;
        char ch;
        int num;
        loc = start;
        do
        {
                end = loc;
                loc = loc->link;
        } while(loc!=start);

        printf("\n\n\t Enter the number you want to delete:");
        scanf("%d",&num);
        i = search(start,num);
        if(i== NULL)
        {
                printf("\n\n\t This number does not exist in list");
                getch();
                return start;
        }

        temp = start;
        do
        {
                if(tem==start && temp->link==start && temp->data==num)
                {
```

```
                free(start);
                start = NULL;
                break;
            }
            else if(temp==start && temp->data==num)
            {
                ptr=start;
                start=start->link;
                end->link = start;
                free(ptr);
                break;
            }
            else if(temp->link->data==num)
            {
                ptr = temp->link;
                temp->link = temp->link->link;
                free(ptr);
                break;
            }
            else  if(temp  ->link->link  ==  start  &&  temp->link-
        >data==num) {
                ptr= temp->link;
                temp->link = start;
                free(ptr);
            }
            temp = temp->link;
        } while(temp!=start);
        return(start);
}
```