DS & ALGORITHMS ASSIGNMENT 1

NAME:MOHAMMED AMRATH **ID**:B181094 **CLASS**:AB2-305

1. Write a program that uses functions to perform the following operations on:

SINGLE LINKED LIST

i) Create a singly linked list
ii) Insert a new node at the beginning of a Singly linked list.
iii) Insert a new node at the given position of a singly linked list.
iv) Insert a new node at the end of a Singly linked list.
v) Delete first node of a singly linked list.
vi) Delete a node from the given position of a singly linked list.
vii) Delete the last node of a singly linked list.
viii) Count the number of nodes in the list

Pseudocode

Create a struct variable
    Struct node
    Int data
    Struct node *link
struct node *root=NULL;

Insert at Beginning

Create a node
Enter the data and store in node->data
node link is NULL
if root==NULL
    root = node
else
    node->link = root
    root = node

Insert at Given position

Take the position as input
Create two struct pointer *p,*q
Int I
P= root
Q=p->link
For(i=1;i<pos-1;i++)
    P=p->Link
    Q=q->link
Node->link=q
p->link=node

Insert at end

Create temp

Temp=root
While(temp->link!=NULL)
    Temp=temp->link
Temp->link =node

Delete First Node

Create *temp;
    temp=root;
    root=root->link;
    free(temp);

Delete Last Node

Create two pointers *p *q
P=root
Q=p->link
While(q->link!=NULL)
    P=p->link
    Q=q->link
Store q in p->link
Free(q)

Delete at given position

Input the position
Create two struct pointers *P,*q;
P=root
Q=p->link
For(i=1;i<pos-1;i++)
    P=p->link
    Q=q->link
p->link = q->link
q->Link=NULL
free(q)

Length of node

Int count=0
Create struct pointer temp
Store root in temp
While(temp!=NULL)
    Count++
    Temp=temp->link;
Return count

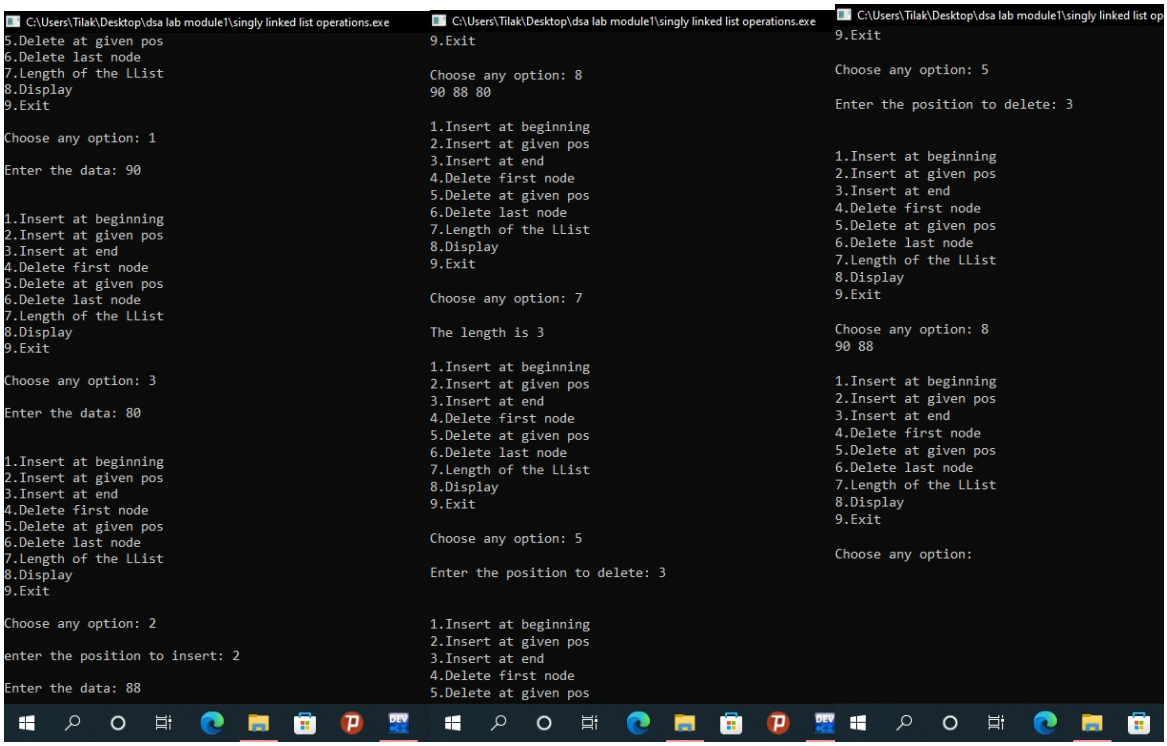
CODE FOR SINGLY LINKED LIST OPERATIONS

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

struct node {
    int data;
```

```
	struct node *link;
};
struct node *root=NULL;

void InsertAtBeg(void)
{
	struct node *newnode,*temp;
	newnode=(struct node* )malloc(sizeof(struct node));
	printf("\nEnter the data: ");
	scanf("%d",&newnode->data);
	newnode->link=NULL;
	if(root==NULL){
		root=newnode;
	}
	else{
		newnode->link=root;
		root=newnode;
	}
}

void InsertAtPos(void)
{
	struct node *newnode,*p,*q;
	int pos;
	newnode=(struct node* )malloc(sizeof(struct node));
	printf("\nenter the position to insert: ");
	scanf("%d",&pos);
	printf("\nEnter the data: ");
	scanf("%d",&newnode->data);
	newnode->link=NULL;
	if(root==NULL){
		root=newnode;
	}
	else{
		int i;
		p=root;
		q=p->link;
		for(i=1;i<pos-1;i++){
			p=p->link;
			q=q->link;
		}
		newnode->link=q;
		p->link=newnode;

	}
}

void InsertAtEnd(void)
{
	struct node *newnode,*temp;
	newnode=(struct node* )malloc(sizeof(struct node));
	printf("\nEnter the data: ");
	scanf("%d",&newnode->data);
	newnode->link=NULL;
	if(root==NULL){
		root=newnode;
	}
	else{
		temp=root;
		while(temp->link!=NULL){
			temp=temp->link;
		}
		temp->link=newnode;
```

```
	}
}

void DeleteFirstNode(void)
{
	struct node *temp;
	temp=root;
	root=root->link;
	free(temp);
}


void DeleteLastNode(void)
{
	struct node *p,*q;
	p=root;
	q=p->link;
	if(root==NULL){
		printf("\nList is empty");
	}
	else{
		if(root->link==NULL){
			root=NULL;
		}
		else{
			while(q->link!=NULL){
				p=p->link;
				q=q->link;
			}
			p->link=NULL;
			free(q);
		}
	}
}

int Length(void)
{
	int count=0;
	struct node *temp;
	temp=root;
	while(temp!=NULL){
		count++;
		temp=temp->link;
	}
	return count;
}

void DeletePosNode(void)
{
	struct node *p,*q;
	int pos;
	printf("\nEnter the position to delete: ");
	scanf("%d",&pos);
	if(pos==1){
		DeleteFirstNode();
	}
	else if(pos==Length()){
		DeleteLastNode();
	}
	else{
		p=root;
		q=p->link;
		int i;
```

```
                for(i=1;i<pos-1;i++){
                        p=p->link;
                        q=q->link;
                }
                p->link=q->link;
                q->link=NULL;
                free(q);
        }
}

void display(void)
{
        struct node *temp;
        if(root==NULL){
                printf("\nList is empty");
        }
        else{
                temp=root;
                while(temp!=NULL){
                        printf("%d ",temp->data);
                        temp=temp->link;
                }
        }
}


int main()
{
        int ch,len;
        printf("Singly linked list operations");
        while(1){
                printf("\n\n1.Insert at beginning");
                printf("\n2.Insert at given pos");
                printf("\n3.Insert at end");
                printf("\n4.Delete first node");
                printf("\n5.Delete at given pos");
                printf("\n6.Delete last node");
                printf("\n7.Length of the LList");
                printf("\n8.Display");
                printf("\n9.Exit");
                printf("\n\nChoose any option: ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: InsertAtBeg();break;
                        case 2: InsertAtPos();break;
                        case 3: InsertAtEnd();break;
                        case 4: DeleteFirstNode();break;
                        case 5: DeletePosNode();break;
                        case 6: DeleteLastNode();break;
                        case 7: len=Length(); printf("\nThe length is %d",len);break;
                        case 8: display();break;
                        case 9: exit(1); break;
                        default: printf("\nChoose a valid option");
                }
        }
}
```



2. Write a program that uses functions to perform the following operations on **Doubly linked list**

i) Create a doubly linked list
ii) Insert a new node at the beginning of a doubly linked list.
iii) Insert a new node at the given position of a doubly linked list.
iv) Insert a new node at the end of a doubly linked list.
v) Delete first node of a doubly linked list.
vi) Delete a node from the given position of a doubly linked list.
vii) Delete the last node of a doubly linked list.
viii) Count the number of nodes in the list

Pseudocode

create struct node
    int data
    struct node *left, *right
struct node *root=NULL

Insert at beginning

Create a newnode
Input newnode->data
Newnode->left = newnode->right=NULL
If root is NULL
    Root=newnode
Else
    newnode->right=root;

root->left=newnode;
root=newnode;

Insert at End

Create a newnode
Struct node *temp
temp=root;
while(temp->right!=NULL)
temp=temp->right;
temp->right=newnode; newnode->left=temp;

Insert at given position

Input the position
Create a newnode
Struct node *p,*q
P=root
Q=p->right;
For(i=1;i<pos-1;i++)
P=p->right
Q=q->right
newnode->right=q
q->left=newnode
newnode->left=p
p->right=newnode

Delete First Node

If there is one node
If(root->left==NULL && root->right==NULL)
Root=NULL
Free(root)
Else
Temp=root
Root=root->right
Root->left=Temp->right=NULL
Free(temp)

Delete Last Node

P=root
Q=p->right
Traverse the q pointer to last node
While(q->right!=NULL)
P=p->right
Q=q->right
p->right=NULL
free(q)

Delete node at given position

```
Struct node *p,*q;
Input position
P=root
Q=p->right
Int i
For(i=1;I<pos-1;i++)
        P=p->right
        Q=q->right
p->right=q->right
q->right=q->left=NULL
free(q)
```

Number of nodes

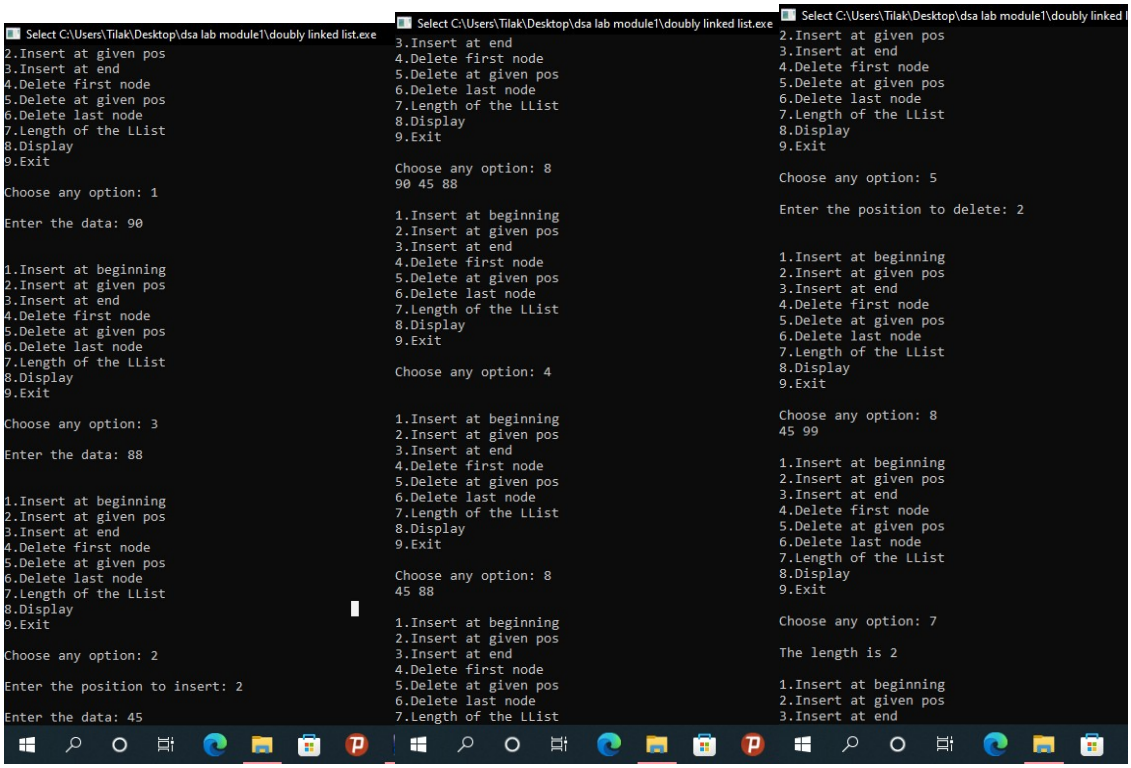
```
Int count=0
temp=root;
while(temp!=NULL)
        count++;
        temp=temp->right;
return count;
#include<stdio.h>
#include<stdlib.h>

struct node {
        int data;
        struct node *left;
        struct node *right;
};
struct node *root=NULL;

void InsertAtBeg(void)
{
        struct node *newnode;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("\nEnter the data: ");
        scanf("%d",&newnode->data);
        newnode->left=newnode->right=NULL;
        if(root==NULL){
                root=newnode;
        }
        else{
                newnode->right=root;
                root->left=newnode;
                root=newnode;
        }
```

```
}

void InsertAtPos(void)
{
	struct node *newnode,*p,*q;
	int pos;
	newnode=(struct node*)malloc(sizeof(struct node));
	printf("\nEnter the position to insert: ");
	scanf("%d",&pos);
	newnode->left=newnode->right=NULL;
	if(pos==1){
		InsertAtBeg();
	}
	else{
		printf("\nEnter the data: ");
		scanf("%d",&newnode->data);
		int i;
		p=root;
		q=p->right;
		for(i=1;i<pos-1;i++){
			p=p->right;
			q=q->right;
		}
		newnode->right=q;
		q->left=newnode;
		newnode->left=p;
		p->right=newnode;
	}
}

void InsertAtEnd(void)
{
	struct node *newnode,*temp;
	newnode=(struct node*)malloc(sizeof(struct node));
	printf("\nEnter the data: ");
	scanf("%d",&newnode->data);
	newnode->left=newnode->right=NULL;
	if(root==NULL){
		root=newnode;
	}
	else{
		temp=root;
		while(temp->right!=NULL){
			temp=temp->right;
		}
```

```
                temp->right=newnode;
                newnode->left=temp;
        }
}

void DeleteFirstNode(void)
{
        struct node *temp;
        if(root->left==NULL && root->right==NULL){
                root=NULL;
                free(root);
        }
        else{
                temp=root;
                root=root->right;
                root->left=NULL;
                temp->right=NULL;
                free(temp);
        }
}

void DeleteLastNode(void)
{
        struct node *p,*q;
        p=root;
        q=p->right;
        if(root==NULL){
                printf("\nList is empty");
        }
        else{
                if(root->right==NULL && root->left==NULL){
                        root=NULL;
                }
                else{
                        while(q->right!=NULL){
                                p=p->right;
                                q=q->right;
                        }
                        p->right=NULL;
                        free(q);
                }
        }
}

int Length(void)
```

```
{
	int count=0;
	struct node *temp;
	temp=root;
	while(temp!=NULL){
		count++;
		temp=temp->right;
	}
	return count;
}

void DeletePosNode(void)
{
	struct node *p,*q;
	int pos;
	printf("\nEnter the position to delete: ");
	scanf("%d",&pos);
	if(pos==1){
		DeleteFirstNode();
	}
	else if(pos==Length()){
		DeleteLastNode();
	}
	else{
		p=root;
		q=p->right;
		int i;
		for(i=1;i<pos-1;i++){
			p=p->right;
			q=q->right;
		}
		p->right=q->right;
		q->right=q->left=NULL;
		free(q);
	}
}

void display(void)
{
	struct node *temp;
	if(root==NULL){
		printf("\nList is empty");
	}
	else{
		temp=root;
```

```
        while(temp!=NULL){
            printf("%d ",temp->data);
            temp=temp->right;
        }
    }
}


int main()
{
    int ch,len;
    printf("Doubly linked list operations");
    while(1){
        printf("\n\n1.Insert at beginning");
        printf("\n2.Insert at given pos");
        printf("\n3.Insert at end");
        printf("\n4.Delete first node");
        printf("\n5.Delete at given pos");
        printf("\n6.Delete last node");
        printf("\n7.Length of the LList");
        printf("\n8.Display");
        printf("\n9.Exit");
        printf("\n\nChoose any option: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: InsertAtBeg();break;
            case 2: InsertAtPos();break;
            case 3: InsertAtEnd();break;
            case 4: DeleteFirstNode();break;
            case 5: DeletePosNode();break;
            case 6: DeleteLastNode();break;
            case 7: len=Length(); printf("\nThe length is %d",len);break;
            case 8: display();break;
            case 9: exit(1); break;
            default: printf("\nChoose a valid option");
        }
    }
}
```



3. Write a program that uses functions to perform the following operations on **Circular singly linked list**

i) Create a circular singly linked list
ii) Insert a new node at the beginning of a circular Sigly linked list.
iii) Insert a new node at the given position of a circular singly linked list.
iv) Insert a new node at the end of a circular Sigly linked list.
v) Delete first node of a circular singly linked list.
vi) Delete a node from the given position of a circular singly linked list.
vii) Delete the last node of a circular singly linked list.
viii) Count the number of nodes in the list

Pseudocode

Struct node
    Int data
    Struct node *link

Insert at beginning

create a newnode
input data
assign to newnode->data
if(root==NULL)
    root=newnode
    newnode->link=root
else
    traverse the temp to last and store the newnode address

```
        temp=root
        while(temp->link!=root)
                temp=temp->link
        temp->link=newnode;
        newnode->link=root;
        root=newnode;
```

Insert at End

Traverse the temp to last node

```
Temp=root
While(temp->link!=root)
        Temp=temp->link
Temp->Link=newnode
Newnode->Link=root
```

Insert At given Position

```
Struct node *p,*q;
Int pos,I;
P=root
Q=p->link
For(i=1;i<pos-1;i++)
        P=p->link
        Q=q->link
Newnode->link=q
p->link=newnode
```

Delete First Node

Traverse the temp to last node and store the root->link address

```
Temp=p=root
While(temp->link!=root)
        Temp=temp->link
Temp->link=root-Link;
Root=root->link
Free(p)
```

Delete Last Node

```
Struct node *p,*q
P=root
Q=p->link
While(q->link!=root)
        Q=q->link
        P=p->link
p->link= root;
q->link=null
free(q)
```

Delete at given position

```
Struct node *p,*q
Input the position to delete
p=root;
q=p->link
int i
for(i=1;i<pos-1;i++)
        p=p->link
        q=q->link
p->link=q->lin
q->link=NULL;
```

Number of nodes

```
Int count=0
Struct node *temp
Temp=root
While(temp->link !=root)
        Count++
        Temp=temp->link
Count++
Return count
```

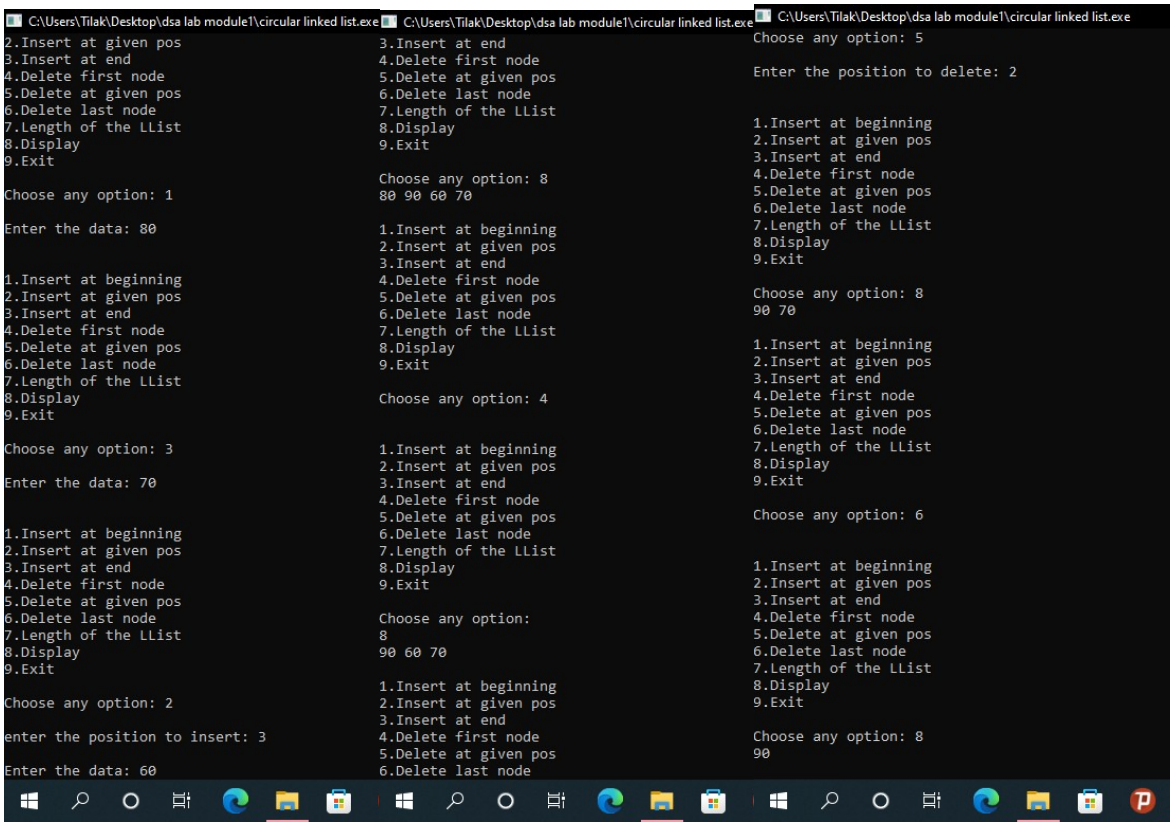
Circular Linked list code

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

struct node {
        int data;
        struct node *link;
};
struct node *root=NULL;

void InsertAtBeg(void)
{
        struct node *newnode,*temp;
        newnode=(struct node* )malloc(sizeof(struct node));
        printf("\nEnter the data: ");
        scanf("%d",&newnode->data);
        if(root==NULL){
                root=newnode;
                newnode->link=root;
        }
        else{
                temp=root;
                while(temp->link!=root){
                        temp=temp->link;
                }
                temp->link=newnode;
                newnode->link=root;
                root=newnode;
        }
}

void InsertAtEnd(void)
```

```
{
	struct node *newnode,*temp;
	newnode=(struct node* )malloc(sizeof(struct node));
	printf("\nEnter the data: ");
	scanf("%d",&newnode->data);
	newnode->link=NULL;
	if(root==NULL){
		root=newnode;
		newnode->link=root;
	}
	else{
		temp=root;
		while(temp->link!=root){
			temp=temp->link;
		}
		temp->link=newnode;
		newnode->link=root;
	}
}

int Length(void)
{
	int count=0;
	struct node *temp;
	temp=root;
	if(temp==NULL){
		return count;
	}
	else{
		while(temp->link!=root){
			count++;
			temp=temp->link;
		}
		count++;
		return count;
	}
}

void InsertAtPos(void)
{
	struct node *newnode,*p,*q;
	int pos;
	newnode=(struct node* )malloc(sizeof(struct node));
	printf("\nenter the position to insert: ");
	scanf("%d",&pos);
	if(pos==1){
		InsertAtBeg();
	}
	else{
		printf("\nEnter the data: ");
		scanf("%d",&newnode->data);
		newnode->link=NULL;
		int i;
		p=root;
		q=p->link;
		for(i=1;i<pos-1;i++){
			p=p->link;
			q=q->link;
		}
		newnode->link=q;
		p->link=newnode;

	}
```

```
}

void DeleteFirstNode(void)
{
	struct node *temp,*p;
	p=temp=root;
	if(root==NULL){
		printf("\nThe list is empty");
	}
	else{
		if(root->link==root){ //if one node is there
			root=NULL;
			free(root);
		}
		else{
			while(temp->link!=root){
				temp=temp->link;
			}
			temp->link=root->link;
			root=root->link;
			free(p);
		}
	}

}


void DeleteLastNode(void)
{
	struct node *p,*q;
	if(root==NULL){
		printf("\nList is empty");
	}
	else{
		if(root->link==root){ //if one node is there
			root=NULL;
			free(root);
		}
		else{
			p=root;
			q=p->link;
			while(q->link!=root){
				q=q->link;
				p=p->link;
			}
			p->link=root;q->link=NULL;
			free(q);
		}

	}
}

void DeletePosNode(void)
{
	struct node *p,*q;
	int pos;
	printf("\nEnter the position to delete: ");
	scanf("%d",&pos);
	if(pos==1){
		DeleteFirstNode();
	}
	else if(pos==Length()){
		DeleteLastNode();
```

```
	}
	else{
		p=root;
		q=p->link;
		int i;
		for(i=1;i<pos-1;i++){
			p=p->link;
			q=q->link;
		}
		p->link=q->link;
		q->link=NULL;
		free(q);
	}
}

void display(void)
{
	struct node *temp;
	if(root==NULL){
		printf("\nList is empty");
	}
	else{
		temp=root;
		while(temp->link!=root){
			printf("%d ",temp->data);
			temp=temp->link;
		}
		printf("%d",temp->data);
	}
}
int main()
{
	int ch,len;
	printf("Circular linked list operations");
	while(1){
		printf("\n\n1.Insert at beginning");
		printf("\n2.Insert at given pos");
		printf("\n3.Insert at end");
		printf("\n4.Delete first node");
		printf("\n5.Delete at given pos");
		printf("\n6.Delete last node");
		printf("\n7.Length of the LList");
		printf("\n8.Display");
		printf("\n9.Exit");
		printf("\n\nChoose any option: ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1: InsertAtBeg();break;
			case 2: InsertAtPos();break;
			case 3: InsertAtEnd();break;
			case 4: DeleteFirstNode();break;
			case 5: DeletePosNode();break;
			case 6: DeleteLastNode();break;
			case 7: len=Length(); printf("\nThe length is %d",len);break;
			case 8: display();break;
			case 9: exit(1); break;
			default: printf("\nChoose a valid option");
		}
	}
}
```



4. Write a program that uses functions to perform the following operations on **Circular doubly linked list**

i) Create a circular doubly linked list
ii) Insert a new node at the beginning of a circular doubly linked list.
iii) Insert a new node at the given position of a circular doubly linked list.
iv) Insert a new node at the end of a circular doubly linked list.
v) Delete first node of a circular doubly linked list.
vi) Delete a node from the given position of a circular doubly linked list.
vii) Delete the last node of a circular doubly linked list.
viii) Count the number of nodes in the list

Pseudocode

Struct node
    Int data
    Struct node *right,*left
Create root node
Insert at Beginning
If root ==null
    Root=newnode
    Newnode->right=newnode->left=root
Else
    Temp=root
    Traverse temp to last node

```
        Temp->right =newnode
        Newnode->right=root
        Root=newnode
```

Insert at End

```
Temp=root
While(temp->right!=NULL)
        Temp=temp->right
Temp->right=newnode
Newnode->left=temp
Newnode->right=root
```

Insert at given position

```
Take two struct pointers *p,*q
P=root
Q=p->right
While(q->right!=NULL)
        P=p->right; q=q->right
newnode->right=q
q->left=newnode
newnode->left=p
p->right=newnode;
```

Delete First Node

```
Temp=root
While(temp->right!=root)
        Temp=temp->right
Temp->right=root->right
Root=root->right
```

Delete Last Node

```
P=root
Q=p->right
While(q->right!=root)
        P=p->right
        Q=q->right
p->right=root
free q
```

Delete at given position

```
Take two pointers *P,*q
P=root
Q=p->right
Int pos,i
For(i=1; i<pos-1;i++)
        P=p->right
```

Q=q->right
p->right=q->right
q->right=q->left=NULL
free q

Number of nodes

Int count=0
If root->right=root->left=root
Count=1
Else
While(Temp->right!=root)
Temp=temp->right
Count++
Count++
Return count

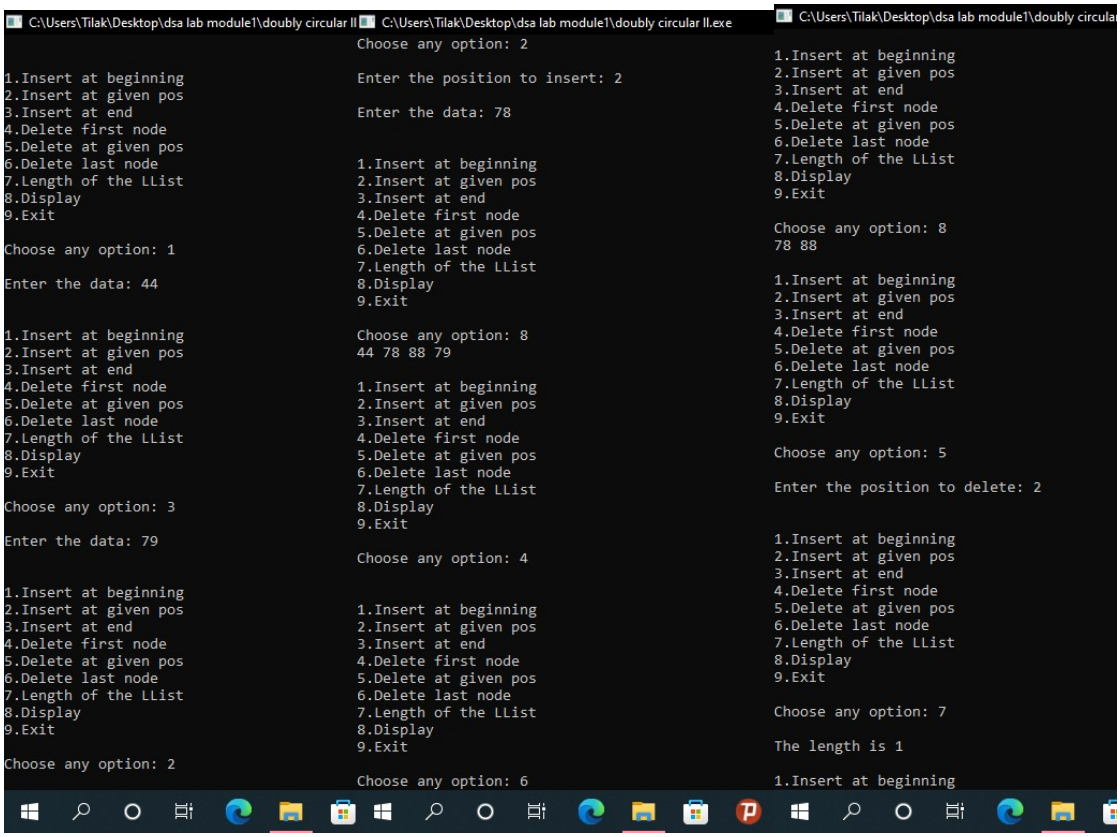
Doubly circular linked list code

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

struct node {
	int data;
	struct node *left;
	struct node *right;
};
struct node *root=NULL;

void InsertAtBeg(void)
{
	struct node *newnode,*temp;
	newnode=(struct node*)malloc(sizeof(struct node));
	printf("\nEnter the data: ");
	scanf("%d",&newnode->data);
	newnode->left=newnode->right=NULL;
	if(root==NULL){
		root=newnode;
		newnode->left=newnode->right=root;
	}
	else{
		temp=root;
		while(temp->right!=root){
			temp=temp->right;
		}
		temp->right=newnode;
		newnode->right=root;
		root->left=newnode;
		root=newnode;
	}
}

void InsertAtPos(void)
{
	struct node *newnode,*p,*q;
	int pos;
	newnode=(struct node*)malloc(sizeof(struct node));
	printf("\nEnter the position to insert: ");
	scanf("%d",&pos);
	newnode->left=newnode->right=NULL;
```

```
	if(pos==1){
		InsertAtBeg();
	}
	else{
		printf("\nEnter the data: ");
		scanf("%d",&newnode->data);
		int i;
		p=root;
		q=p->right;
		for(i=1;i<pos-1;i++){
			p=p->right;
			q=q->right;
		}
		newnode->right=q;
		q->left=newnode;
		newnode->left=p;
		p->right=newnode;
	}
}

void InsertAtEnd(void)
{
	struct node *newnode,*temp;
	newnode=(struct node*)malloc(sizeof(struct node));
	printf("\nEnter the data: ");
	scanf("%d",&newnode->data);
	if(root==NULL){
		root=newnode;
		newnode->right=newnode->left=root;
	}
	else{
		temp=root;
		while(temp->right!=root){
			temp=temp->right;
		}
		temp->right=newnode;
		newnode->left=temp;
		newnode->right=root;
	}
}

void DeleteFirstNode(void)
{
	struct node *temp,*p;
	if(root->left==root && root->right==root){
		free(root);
		root=NULL;
	}
	else{
		p=temp=root;
		while(temp->right!=root){
			temp=temp->right;
		}
		temp->right=root->right;
		root=root->right;
		free(p);
	}
}

void DeleteLastNode(void)
{
	struct node *p,*q;
	p=root;
```

```
	q=p->right;
	if(root==NULL){
		printf("\nList is empty");
	}
	else{
		if(root->right==root && root->left==root){
			free(root);
			root=NULL;
		}
		else{
			while(q->right!=root){
				p=p->right;
				q=q->right;
			}
			p->right=root;
			free(q);
		}
	}
}

int Length(void)
{
	int count=0;
	struct node *temp;
	temp=root;
	if(root->right==root){
		count++;return count;
	}
	else{
		while(temp->right!=root){
			count++;
			temp=temp->right;
		}
		count++;
		return count;
	}
}

void DeletePosNode(void)
{
	struct node *p,*q;
	int pos;
	printf("\nEnter the position to delete: ");
	scanf("%d",&pos);
	if(pos==1){
		DeleteFirstNode();
	}
	else if(pos==Length()){
		DeleteLastNode();
	}
	else{
		p=root;
		q=p->right;
		int i;
		for(i=1;i<pos-1;i++){
			p=p->right;
			q=q->right;
		}
		p->right=q->right;
		q->right=q->left=NULL;
		free(q);
	}
}
```

```
void display(void)
{
	struct node *temp;
	if(root==NULL){
		printf("\nList is empty");
	}
	else{
		temp=root;
		while(temp->right!=root){
			printf("%d ",temp->data);
			temp=temp->right;
		}
		printf("%d",temp->data);
	}
}


int main()
{
	int ch,len;
	printf("Doubly Circular linked list operations");
	while(1){
		printf("\n\n1.Insert at beginning");
		printf("\n2.Insert at given pos");
		printf("\n3.Insert at end");
		printf("\n4.Delete first node");
		printf("\n5.Delete at given pos");
		printf("\n6.Delete last node");
		printf("\n7.Length of the LList");
		printf("\n8.Display");
		printf("\n9.Exit");
		printf("\n\nChoose any option: ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1: InsertAtBeg();break;
			case 2: InsertAtPos();break;
			case 3: InsertAtEnd();break;
			case 4: DeleteFirstNode();break;
			case 5: DeletePosNode();break;
			case 6: DeleteLastNode();break;
			case 7: len=Length(); printf("\nThe length is %d",len);break;
			case 8: display();break;
			case 9: exit(1); break;
			default: printf("\nChoose a valid option");
		}
	}
}
```


C:\Users\Tilak\Desktop\dsa lab module1\doubly circular ll
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 1
Enter the data: 44
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 3
Enter the data: 79
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 2
C:\Users\Tilak\Desktop\dsa lab module1\doubly circular ll.exe
Choose any option: 2
Enter the position to insert: 2
Enter the data: 78
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 8
44 78 88 79
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 4
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 6
C:\Users\Tilak\Desktop\dsa lab module1\doubly circula
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 8
78 88
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 5
Enter the position to delete: 2
1.Insert at beginning
2.Insert at given pos
3.Insert at end
4.Delete first node
5.Delete at given pos
6.Delete last node
7.Length of the LList
8.Display
9.Exit
Choose any option: 7
The length is 1
1.Insert at beginning