DATE : 26/11/2024

Circular Linked List

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

struct Node{
        int data;
        struct Node * link;
};

typedef struct Node node;
node *start = NULL;
node *new1, *curr1, *ptr;

void create();
void display();
void InsertStart();
void InsertPosition();
void InsertEnd();
void DeleteStart();
void DeletePosition();
void DeleteEnd();

void main(){
        int ch;
        while(1){
            printf("\n1. Create \n2. Display \n3. Insert at Beginning \n4. Insert
              at Position \n5. Insert at End \n6. Delete from Beginning \n7.
              ~~Insert~~ Delete at Position \n8. Delete at End \n9. Exit");
        printf("\n Enter Your Choice: ");
        scanf("%d", &ch);
```

```
            switch (ch){
                case 1: create();
                        break;
                case 2: display();
                        break;
                case 3: InsertStart();
                        break;
                case 4: InsertPosition();
                        break;
                case 5: InsertEnd();
                        break;
                case 6: DeleteStart();
                        break;
                case 7: DeletePosition();
                        break;
                case 8: DeleteEnd();
                        break;
                case 9: exit(0);
            }
        }
}

void create(){
        char ch;
        do{
            new1=(node *)malloc(sizeof(node));
            printf("\n Enter Value: ");
            scanf("%d", &new1->data);
            if (start == NULL){
                start=new1;
                new1->link=start;
            }
        else{
            curr = start;
            while( curr -> link != start){
                    curr = curr -> link;
            }
            curr -> link = new1;
            new1 -> link = start;
        }
        printf("Do You Want to Add an Element (Y/N) ? ");
        scanf(" %d", &ch);
      }while(ch == 'y'||ch == 'Y');
}


void display(){
        if(start == NULL){
            printf("\n Linked List is Empty ");
            return;
        }
        ptr = start;
        printf(" Elements in Circular Linked List : \n");
        do{
            printf("%d  ", ptr -> data);
            ptr = ptr -> link;
      }while(ptr != start);
     printf(" \n");
}


void InsertStart(){
        new1 = (node *)malloc (sizeof (node));
        printf("\n Enter Value: ");
        scanf("%d", &new1 -> data);
```

PAGE NO :
DATE :

```
        if (start == NULL){
                start = new1;
                new1-> link = start;
        }
        else{
                new1 -> link = start;
                start = new1;
                ptr = start;
                while (ptr -> link != start){
                        ptr = ptr -> link;
                }
                ptr -> link = start;
        }
}

void InsertEnd(){
        new1 = (node *)malloc(sizeof(node));
        printf ("\n Enter Value: ");
        scanf("%d", &new1 -> data);

        if (start == NULL){
                start = new1;
                new1-> link = start;
        }
        else{
                curr = start;
                while (curr -> link != start){
                        curr = curr -> link;
                }
                curr -> link = new1;
                new1-> link = start;
        }
}
```

```
void InsertPosition(){
        int i=1, pos;
        new1= (node *)malloc (sizeof (node));
        printf("\n Enter Value: ");
        scanf("%d", &new 1->data);

      if (start == NULL){
            start = new1;
            new 1-> link = start;
            return;
      }
      printf("Enter Position:");
      scanf("%d", &pos);
      if (pos == 1){
            new 1 -> link = start;
            start = new1;
            ptr = start;
            while( ptr -> link != start){
                  ptr = ptr -> link;
            }
            ptr -> link = start;
            return;
      }

     ptr = start;
     while( ptr -> link != start && i< pos - 1){
           ptr = ptr -> link;
           i++;
     }
     if (i==pos-1){
           new1 -> link = ptr -> link;
           ptr -> link = new1;
    }
```

```
        else{
            printf("Position Not Found");
        }
}


void DeleteStart (){
        if (start==NULL){
              printf("\n Linked List is Empty ");
              return;
        }
        node * temp =start;
        if (start -> link ==start){
                start =NULL;
        }
        else{
              ptr= start;
              while (ptr -> link != start){
                     ptr= ptr -> link;
              }
              start = start -> link;
              ptr -> link= start;
        }
        free (temp);
        printf("\n First Element Deleted ");
}


void DeletePosition (){
       int i= 1,pos;
       if (start == NULL){
              printf("\n Linked List is Empty ");
              return;
       }
```

```
        printf("Enter Position ");
        scanf("%d", &pos);
        if (pos==1){
                Deletestart();
                return;
        }
        ptr = start;
        node *prev= NULL;
        while( ptr != start && i<pos){
                prev = ptr;
                ptr = ptr -> link;
                i++;
        }
        if (ptr == start){
              printf(" Position Not Found");
              return;
        }
        prev-> link =ptr -> link;
        free (ptr);
        printf("Element at Position %d Deleted \n");
}

void DeleteEnd(){
        if (Start==NULL){
              printf("Linked List is Empty ");
              return;
        }
        node * temp = start;
        if (start -> link == start){
              start = NULL;
        }
        else{
                ptr= start;
                while(ptr->link != start){
                        ptr = ptr -> link;
                }

                ptr -> link = start;
        }

        free(temp);
        printf("Last Element Deleted");
}
```

Output

```
1. Create
2. Display
3. Insert at Beginning
4. Insert at Position
5. Insert at End
6. Delete from Beginning
7. Delete at Position
8. Delete at End
9. Exit

-> Enter Your Choice: 1
Enter Value: 10
Do You Want to Add an Element(Y/N)? y
Enter Value: 20
Do You Want to Add an Element (Y/N)? y
Enter Value: 30
Do You Want to Add an Element (Y/N)? n

-> Enter Your Choice: 2
Elements in Circular Linked List:
```

```
10   20   30

→ Enter Your Choice : 4
Enter Value : 40
Enter Position : 2

→ Enter Your Choice : 2
Elements in Circular Linked List :
10   40   20   30

→ Enter Your Choice : 5
Enter Value : 50

→ Enter Your Choice : 2
Elements in Circular Linked List
10  40  20  30  50

→ Enter Your Choice : 6
First Element Deleted

→ Enter Your Choice : 2
Elements in Circular Linked List :
40  20  30  50

→ Enter Your Choice : 8
Last Element Deleted

→ Enter Your Choice : 9
```

[illegible]
3/12/24