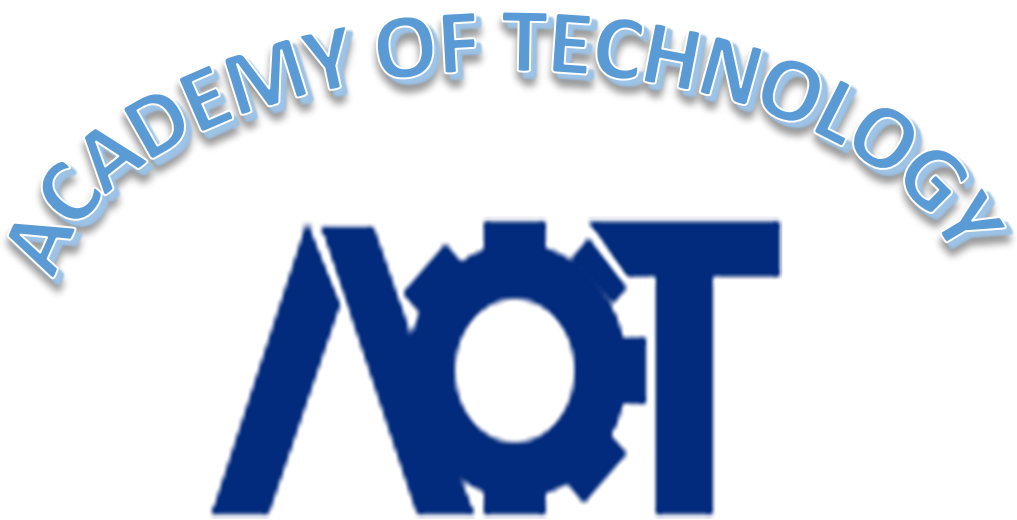

NAME : PRIYADARSHAN GHOSH

COLLEGE ROLL NO: 72

UNIVERSITY ROLL NO: 16900319072

DEPARTMENT: ECE-1(Y)

SEMESTER: 3rd

PAPER CODE : ES-CS391

➢ Laboratory *Assignment #14*

Q1.CREATION OF SINGLY CIRCULAR LINKED LIST,DISPLAY OF SINGLY CIRCULAR LINKED LIST,INSERT A NODE IN DIFFERENT POSITIONS OF SINGLY CIRCULAR LINKED LIST,DELETE A NODE FROM DIFFERENT POSITIONS OF SINGLY CIRCULAR LINKED LIST.

Ans:

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

struct node {
int data;
struct node * next;
}*head;

void createList(int n);
void displayList();
void insert_beginning(int data);
void search_element(int data);
void updating_element(int data);
void insert_given_position(int data, int position);
void delete_beginning();
void delete_given_position();
void reverse_list();
```

```
int main()
{
int n, data, choice=1;
head = NULL;
while(choice != 0)
{

printf("1. Create List\n");
printf("2. Insert at beginning\n");
printf("3. Insert at any position\n");
printf("4. Delete at beginning\n");
printf("5. Delete at any position\n");
printf("6. Search Element\n");

printf("0. Exit\n");
printf("\n\n");
printf("Enter your choice : ");
scanf("%d", &choice);
switch(choice)
{
case 1:
printf("Enter the total number of nodes in list: ");
scanf("%d", &n);
createList(n);
displayList();
```

```
break;
case 2:
printf("Enter data to be inserted at beginning: ");
scanf("%d", &data);
insert_beginning(data);
displayList();
break;
case 3:
printf("Enter node position: ");
scanf("%d", &n);
printf("Enter data you want to insert at %d position: ", n);
scanf("%d", &data);
insert_given_position(data, n);
displayList();
break;
case 4:
if(head == NULL)
{
printf("\nThe list is empty\n");
}
else
{
delete_beginning();
displayList();
}
break;
```

```
case 5:
if(head == NULL)
{
printf("\nThe list is empty\n");
}
else
{
delete_given_position();
displayList();
}
break;
case 6:
printf("\nEnter the element to be searched : ");
scanf("%d",&data);
search_element(data);
break;

default:
printf("Error! Invalid choice.");
}
printf("\n");
}
return 0;
}

void createList(int n)
```

```
{
int i, data;
struct node *prevNode, *newNode;
if(n >= 1)
{

head = (struct node *)malloc(sizeof(struct node));
printf("Data of node 1 : ");
scanf("%d", &data);
head->data = data;
head->next = NULL;
prevNode = head;



for(i=2; i<=n; i++)
{
newNode = (struct node *)malloc(sizeof(struct node));
printf("Data of node %d : ", i);
scanf("%d", &data);
newNode->data = data;
newNode->next = NULL;

prevNode->next = newNode;
prevNode = newNode;
}
```

```
prevNode->next = head;
}
}

void displayList()
{
struct node *current;
int n = 1;
if(head == NULL)
{
printf("List is empty.\n");
}
else
{
current = head;
printf("The SINGLY CIRCULAR LINKED LIST IS : \n");
do {
printf("%d\t",current->data);
current = current->next;
n++;
}while(current != head);
}
}

void insert_beginning(int data)
{
```

```
struct node *newNode, *current;
if(head == NULL)
{
printf("List is empty.\n");
}
else
{

newNode = (struct node *)malloc(sizeof(struct node));
newNode->data = data;
newNode->next = head;
printf("\nThe element %d is inserted at the beginning",data);
printf("\n");

current = head;
while(current->next != head)
{
current = current->next;
}
current->next = newNode;

head = newNode;
}
}

void insert_given_position(int data, int position)
```

```
{
struct node *newNode, *current;
int i;
if(head == NULL)
{
printf("List is empty.\n");
}
else if(position == 0)
{
insert_beginning(data);
}
else
{

newNode = (struct node *)malloc(sizeof(struct node));
newNode->data = data;
printf("\nThe element %d is inserted at index %d",data,position);
printf("\n");

current = head;
for(i=2; i<=position; i++)
{
current = current->next;
}

newNode->next = current->next;
```

```
current->next = newNode;
}
}

void delete_beginning()
{
struct node * temp,*s;

if (head == head->next)
{
head = NULL;
printf("\nThe List is empty\n");
}
else
{
temp = head;
s = head;
while (temp->next != head)
{
temp = temp -> next;
}
printf("\nThe element %d is deleted at the beginning",s -> data);
printf("\n");
head = s->next;
temp->next = head;
printf("\n");
```

```
free(s);
}
}

void delete_given_position()
{
struct node * temp, *s;
if (head == NULL)
printf("\nThe List is empty");
else
{
int count = 0, pos;
printf("\nEnter the position to be deleted : ");
scanf("%d", &pos);
temp = head;
while (count < pos)
{
s = temp;
temp = temp -> next;
count++;
}
printf("\nThe element %d at index %d is deleted",temp -> data,pos);
printf("\n");
s -> next = temp -> next;
printf("\n");
free(temp);
```

```
}
}

void search_element(int data)
{
struct node * temp = head;
int index = 0;
while(temp)
{
if(temp -> data == data)
{
printf("\nElement found at index %d in the list",index);
break;
}
else
{
temp = temp -> next;
index++;
}
}
}
```

OUTPUT =>

1. Create List
2. Insert at beginning
3. Insert at any position

4. Delete at beginning

5. Delete at any position

6. Search Element

0. Exit

Enter your choice : 1

Enter the total number of nodes in list: 6

Data of node 1 : 8

Data of node 2 : 6

Data of node 3 : 7

Data of node 4 : 9

Data of node 5 : 5

Data of node 6 : 1

The SINGLY CIRCULAR LINKED LIST IS :

8 6 7 9 5 1

1. Create List

2. Insert at beginning

3. Insert at any position

4. Delete at beginning

5. Delete at any position

6. Search Element

0. Exit

Enter your choice : 2

Enter data to be inserted at beginning: 4

The element 4 is inserted at the beginning

The SINGLY CIRCULAR LINKED LIST IS :

4 8 6 7 9 5 1

1. Create List

2. Insert at beginning

3. Insert at any position

4. Delete at beginning

5. Delete at any position

6. Search Element

0. Exit

Enter your choice : 3

Enter node position: 6

Enter data you want to insert at 6 position: 8

The element 8 is inserted at index 6

The SINGLY CIRCULAR LINKED LIST IS :

4 8 6 7 9 5 8 1

1. Create List

2. Insert at beginning

3. Insert at any position

4. Delete at beginning

5. Delete at any position

6. Search Element

0. Exit

Enter your choice : 4

The element 4 is deleted at the beginning

The SINGLY CIRCULAR LINKED LIST IS :

8 6 7 9 5 8 1

1. Create List

2. Insert at beginning

3. Insert at any position

4. Delete at beginning

5. Delete at any position

6. Search Element

0. Exit

Enter your choice : 5

Enter the position to be deleted : 3

The element 9 at index 3 is deleted

The SINGLY CIRCULAR LINKED LIST IS :

```
8      6      7      5      8      1
1. Create List
2. Insert at beginning
3. Insert at any position
4. Delete at beginning
5. Delete at any position
6. Search Element
0. Exit

Enter your choice : 0
--------------------------------
Process exited after 59.64 seconds with return value 0
Press any key to continue . . .
```

Q3.SOLVE THE JOSEPHUS'S PROBLEM USING CIRCULAR LINKED LIST.

Ans:

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

struct node
{
    int num;
    struct node *next;
};
```

```
void create(struct node **);
void display(struct node *);
int survivor(struct node **, int);

int main()
{
    struct node *head = NULL;
    int survive, skip;

    create(&head);
    printf("The persons in circular list are:\n");
    display(head);
    printf("Enter the number of persons to be skipped: ");
    scanf("%d", &skip);
    survive = survivor(&head, skip);
    printf("The person to survive is : %d\n", survive);
    free(head);

    return 0;
}

int survivor(struct node **head, int k)
{
    struct node *p, *q;
    int i;
```

```
    q = p = *head;
    while (p->next != p)
    {
        for (i = 0; i < k - 1; i++)
        {
            q = p;
            p = p->next;
        }
        q->next = p->next;
        printf("%d has been killed.\n", p->num);
        free(p);
        p = q->next;
    }
    *head = p;

    return (p->num);
}

void create (struct node **head)
{
    struct node *temp, *rear;
    int a, ch;

    do
    {
```

```
        printf("Enter a number: ");
        scanf("%d", &a);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = a;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        printf("Do you want to add a number [1/0]? ");
        scanf("%d", &ch);
    } while (ch != 0);
    rear->next = *head;
}

void display(struct node *head)
{
    struct node *temp;

    temp = head;
    printf("%d   ", temp->num);
```

```
    temp = temp->next;
    while (head != temp)
    {
        printf("%d   ", temp->num);
        temp = temp->next;
    }
    printf("\n");
}
```

OUTPUT =>

```
Enter a number: 1
Do you want to add a number [1/0]? 1
Enter a number: 2
Do you want to add a number [1/0]? 1
Enter a number: 3
Do you want to add a number [1/0]? 1
Enter a number: 4
Do you want to add a number [1/0]? 1
Enter a number: 5
Do you want to add a number [1/0]? 1
Enter a number: 6
Do you want to add a number [1/0]? 1
Enter a number: 7
Do you want to add a number [1/0]? 0
The persons in circular list are:
1   2   3   4   5   6   7
```

```
Enter the number of persons to be skipped: 3
3 has been killed.
6 has been killed.
2 has been killed.
7 has been killed.
5 has been killed.
1 has been killed.
The person to survive is : 4
--------------------------------
Process exited after 21.73 seconds with return value 0
Press any key to continue . . .
```