ACADEMY OF TECHNOLOGY

NAME : PRIYADARSHAN GHOSH

COLLEGE ROLL NO: 72

UNIVERSITY ROLL NO: 16900319072

DEPARTMENT: ECE-1(Y)

SEMESTER: 3rd

PAPER CODE : ES-CS391

➢ Laboratory *Assignment #16*

1. WRITE PROGRAMS TO PERFORM FOLLOWING OPERATIONS USING FUNCTIONS:

A. CREATION OF DOUBLY CIRCULAR LINEAR LINKED LIST

B. DISPLAY OF DOUBLY CIRCULAR LINEAR LINKED LIST

C. INSERT A NODE IN DIFFERENT POSITIONS OF DOUBLY CIRCULAR LINEAR LINKED LIST

D. DELETE A NODE FROM DIFFERENT POSITIONS OF DOUBLY CIRCULAR LINEAR LINKED LIST

Ans:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void deletion_beginning();
void deletion_last();
void display();

main ()
{
```

```
int choice =0;
    while(choice != 9)
    {
        printf("\nMain Menu");

        printf("\n1.Insert in Beginning\n2.Insert at last\n3.Delete from
Beginning\n4.Delete from last\n5.Show\n6.Exit\n");
        printf("\nEnter your choice?\n");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
            insertion_beginning();
            break;
            case 2:
                    insertion_last();
            break;
            case 3:
            deletion_beginning();
            break;
            case 4:
            deletion_last();
            break;
            break;
            case 5:
            display();
            break;
            case 6:
```

```
            exit(0);
            break;
            default:
            printf("Please enter valid choice..");
        }
    }
}
void insertion_beginning()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
    printf("\nEnter Item value");
    scanf("%d",&item);
    ptr->data=item;
   if(head==NULL)
   {
      head = ptr;
      ptr -> next = head;
      ptr -> prev = head;
   }
   else
   {
       temp = head;
    while(temp -> next != head)
    {
        temp = temp -> next;
    }
    temp -> next = ptr;
    ptr -> prev = temp;
    head -> prev = ptr;
    ptr -> next = head;
    head = ptr;
   }
   printf("\nNode inserted\n");
}

}
void insertion_last()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *) malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
       printf("\nEnter value");
       scanf("%d",&item);
        ptr->data=item;
       if(head == NULL)
       {
           head = ptr;
           ptr -> next = head;
           ptr -> prev = head;
       }
       else
       {
          temp = head;
          while(temp->next !=head)
          {
              temp = temp->next;
          }
          temp->next = ptr;
          ptr ->prev=temp;
          head -> prev = ptr;
      ptr -> next = head;
        }
   }
     printf("\nnode inserted\n");
}

void deletion_beginning()
```

```
{
    struct node *temp;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        temp = head;
        while(temp -> next != head)
        {
            temp = temp -> next;
        }
        temp -> next = head -> next;
        head -> next -> prev = temp;
        free(head);
        head = temp -> next;
    }

}
void deletion_last()
```

```
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != head)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = head;
        head -> prev = ptr -> prev;
        free(ptr);
        printf("\nnode deleted\n");
    }
}

void display()
```

```
{
    struct node *ptr;
    ptr=head;
    if(head == NULL)
    {
        printf("\nnothing to print");
    }
    else
    {
        printf("\n printing values ... \n");

        while(ptr -> next != head)
        {


            printf("%d\n", ptr -> data);
            ptr = ptr -> next;
        }
        printf("%d\n", ptr -> data);
    }

}
```

OUTPUT =>

```
Main Menu
1.Insert in Beginning
2.Insert at last
```

```
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
1

Enter Item value10

Node inserted

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
1

Enter Item value20

Node inserted
```

```
Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
1

Enter Item value30

Node inserted

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
1

Enter Item value40
```

```
Node inserted

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
2

Enter value50

node inserted

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
```

```
2

Enter value60

node inserted

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
2

Enter value70

node inserted

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
```

```
6.Exit

Enter your choice?
5

 printing values ...
40
30
20
10
50
60
70

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
3

Main Menu
1.Insert in Beginning
```

```
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
3

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
3

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit
```

```
Enter your choice?
3

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
4

node deleted

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
4
```

```
node deleted

Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
4

 UNDERFLOW
Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
5

nothing to print
```

```
Main Menu
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Show
6.Exit

Enter your choice?
6


--------------------------------
Process exited after 64.89 seconds with return value 0
Press any key to continue . . .
```

2A.TO CALCULATE FACTORIAL OF AN INTEGER NUMBER. (TRY TAKING BIG NUMBER ALSO).

Ans:

```
#include<stdio.h>
unsigned long long fact(int n){
    if(n==0) return 1;
    return n*fact(n-1);
}
 main(){
    int n;
```

```c
    printf("Enter the number: ");
    scanf("%d",&n);
    printf("%d! = %llu",n,fact(n));
}
```

OUTPUT =>

```
Enter the number: 34
34! = 4926277576697053184
--------------------------------
Process exited after 6.925 seconds with return value 0
Press any key to continue . . .
```

2.B TO CALCULATE GCD / HCF OF N INTEGER NUMBERS.

Ans:

```c
#include <stdio.h>
int gcd(int a, int b)
{
    if (a == 0)
        return b;
    return gcd(b % a, a);
}
int getGCD(int a[], int n)
{
        int res=a[0];
        for (int i = 1; i < n; i++)
    {
        res = gcd(a[i], res);
        if(res == 1)
            return 1;
    }
    return res;
}
int main()
{
        int n, arr[50], s1, s2;
        printf("Enter range:");
        scanf ("%d", &n);
        printf("Enter elements: ");
        for (int i=0; i<n; i++)
                scanf ("%d", &arr[i]);
        s1 = getGCD(arr, n);
        printf("GCD: %d\n", s1);
}
```

OUTPUT =>

```
Enter range:5
Enter elements: 3 5 7 9 11
GCD: 1


--------------------------------

Process exited after 13.18 seconds with return value 0
Press any key to continue . . .
```

2.C TO GENERATE FIBONACCI SERIES UP TO N TERMS.

Ans:

```
#include<stdio.h>

int Fibonacci(int);

int main()
{
   int n, i = 0, c;
  printf("Enter the number \n");
   scanf("%d",&n);

   printf("Fibonacci series\n");

   for ( c = 1 ; c <= n ; c++ )
   {
      printf("%d\n", Fibonacci(i));
      i++;
   }

   return 0;
}

int Fibonacci(int n)
{
   if ( n == 0 )
```

```
      return 0;
   else if ( n == 1 )
      return 1;
   else
      return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

OUTPUT =>

```
Enter the number
20
Fibonacci series
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
```

```
987

1597

2584

4181


--------------------------------

Process exited after 2.55 seconds with return value 0

Press any key to continue . . .
```

2.D TO SOLVE TOWER OF HANOI PROBLEM FOR DIFFERENT NUMBER OF DISKS.

Ans:

```
#include <stdio.h>

void towers(int, char, char, char);

int main()
{
    int num;

    printf("Enter the number of disks : ");
    scanf("%d", &num);
    printf("The sequence of moves involved in the Tower of Hanoi are
:\n");
    towers(num, 'A', 'C', 'B');
    return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
    if (num == 1)
    {
        printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
        return;
    }
    towers(num - 1, frompeg, auxpeg, topeg);
    printf("\n Move disk %d from peg %c to peg %c", num, frompeg,
topeg);
    towers(num - 1, auxpeg, topeg, frompeg);
}
```

OUTPUT =>

```
Enter the number of disks : 4
The sequence of moves involved in the Tower of Hanoi are :


 Move disk 1 from peg A to peg B
 Move disk 2 from peg A to peg C
 Move disk 1 from peg B to peg C
 Move disk 3 from peg A to peg B
 Move disk 1 from peg C to peg A
 Move disk 2 from peg C to peg B
 Move disk 1 from peg A to peg B
 Move disk 4 from peg A to peg C
 Move disk 1 from peg B to peg C
 Move disk 2 from peg B to peg A
 Move disk 1 from peg C to peg A
 Move disk 3 from peg B to peg C
 Move disk 1 from peg A to peg B
 Move disk 2 from peg A to peg C
 Move disk 1 from peg B to peg C
--------------------------------
Process exited after 2.201 seconds with return value 0
Press any key to continue . . .
```