21ECC212T Data structures And Algorithms

RA2311043010062 Kirthika. KS.

Assignment - 02.

1. Implementation of tower of Hanoi for the 4 disks.

```
#include <stdio.h>
void towerOfHanoi (int n, char from_rod, char to_rod,
                   char aux_rod)
{
 if (n == 1)
 {
  printf ("Move disk 1 from %c to %c \n", from_rod,
          to_rod);
  return;
 }
 towerOfHanoi (n-1, from_rod, aux_rod, to_rod);
 printf ("Move disk from %d to %c to %c \n",
         n, from_rod, to_rod);
 towerOfHanoi (n-1, aux_rod, to_rod, from_rod);
}
int main()
{
  int n = 4;
  tower of Hanoi (n, 'A', 'C', 'B');
  return 0;
```

2. Implementation of circular queue using linked list

```
#include <stdio.h>
#include <stdio.h>
struct Node {
    int data;
    struct Node * next;
};
struct Circular_Queue {
    struct Node* front;
    struct Node* rear;
};
void enqueue (struct Circular_Queue * q, int value) {
    struct Node* temp = (struct Node*)malloc (sizeof (
                        struct Node));

    temp -> data = value;
    temp -> next = NULL;
    if (q -> rear == NULL) {
        q -> front = q -> rear = temp;
        q -> rear -> next = q -> front;
    } else {
        q -> rear -> next = temp;
        q -> rear = temp;
        q -> rear -> next = q -> front;
    }
```

```
void dequeue (struct Circular_Queue * q) {{
    if (q -> front == NULL) {
        printf ("Queue is empty! \n");
        return;
    }
    struct Node * temp = q -> front;
    printf ("Dequeued: %d ", temp -> data);
    if (q -> front == q -> rear) {
        q -> front = q -> rear = NULL;
    } else {
        q -> front = q -> front -> next;
        q -> rear -> next = q -> front;
    }
    free (temp);
}
void display (struct Circular_Queue *q) {
    if (q -> front == NULL) {
        printf ("Queue is empty! \n");
        return;
    }
    struct Node * temp = q -> front;
    printf ("Circular Queue: ");
    do
```

```
do {
    printf("%d ⟷ " temp → data);
    temp = temp → next;
while (temp != last → next));
printf("\n");
return 0;
}
```