Lab-4 11/1/24

1) // Circular Queue

```
#include <stdio.h>
#include <stdlib.h>
#define size 5

int q[size], f=0, r=-1;
int count=0;

void enqueue(int item) {
    if (count == size) {
        printf("\n Queue full!");
        return;
    }
    q[(++r)% size] = item;
    count++;
}

void dequeue() {
    if (count == 0)
        printf("\n Queue empty!");
    f = (f+1) % size;
    count--;
}
```

```
void display(){
    if(count == 0){
        printf("\n Queue empty!");
    }
    int front = f;
    for(int i=0; i<count; i++){
        printf("%d ", q[front]);
        front = (front+1) % size;
    }
}

int main(){
    int ch, item;
    while(1){
        printf("\n Select choice \n 1. Enqueue \n 2. Dequeue \n
                3. Display \n Choice:");
        scanf("%d", &ch);

        switch(ch){
            case 1:
                printf("\n Enter value to insert:");
                scanf("%d", &item);
                enqueue(item);
                break;

            case 2:
                dequeue();
                printf("\n Item popped");
                break;
```

```
        case 3:
            display();
            break;
        exit(0);
    }
  }
}
```

Output:

Select Choice:

1. Enqueue
2. Dequeue
3. Display

choice: 1

Enter value: 3

Queue = 3

---

choice: 1

Enter value = 4

Queue = 3 4

choice: 2

Queue = 4

```
2: // Linked list
#include <stdio.h>
#include<stdlib.h>

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

void insertAtHead(struct node **head, int new){
    struct node *new_node = (struct node*)malloc(sizeof(struct node));

    new->data = new;
    new->next = (*head);

}

void insertatMiddle(struct node *prev_node, int new){
    if(prev == NULL){
        printf("Given previous nod cannot be NULL");
        return;
    }
    struct node *new = (struct node*) malloc(sizeof(struct node));
    new->data = new_dade;
    new->next = prev->next;
    prev->next = new_node;
}
```

```
void insertAtEnd(struct node **head, int new){
    struct node *new = (struct node*) malloc (sizeof(
    struct node *last = *head_ref;

    new_node -> data = new_data;

    if (*head_ref) == NULL){
            *head_ref = new_node;
             return;
    }

    last -> next = new_node;
    return;

}

void display (struct node * node){
    printf("Linked List: ");
    while (node != NULL){
            printf(" %d ", node -> data);
            node = nod -> next;

    }
    printf ("\n");
```

```
t node));
int main(){
    struct node *head =NULL;
    int choice=0, a;
    while(choice!=3){
        printf("\nSelect an option\n1. Insert\n2. Display
                                    \n3. Exit");
        scanf("%d",&choice);
        switch(choice){
            case 1:
              switch(choice2){
                  case1:
                    insertatHead(&head, a);
                  case2;
                    insertinMid(&head, a);
                  case3:
                  insert at End(&hed, a);
              }
            case 2:
                display(head);
        }
    }
```

Output:

Select an option
1. Insert
2. Display
3. Exit
Choice: 1

1: At start
2. At Middle
3. At End.
choice: 1
Enter value: 5
Linked List: 5

choice: 1
1. At start
2. At Middle
3. At End
choice: 3
Enter value: 8
Linked List: 5 8

choice: 2
Linked List: 5 8

Sp.
11/1/24