CIRCULAR QUEUE IMPLEMENTATION USING ARRAY

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5   // Maximum MAX of the circular queue
int cqueue[MAX];
int front = -1, rear = -1;
// Check if queue is full
int isFull() {
   return (front == 0 && rear == MAX - 1);
}
// Check if queue is empty
int isEmpty() {
   return (front == -1);
}
// Insert (Enqueue)
void enqueue(int data) {
   if (isFull()) {
      printf("Queue Overflow! Cannot insert %d\n", data);
      return;
   }
   if (front == -1) { // first element
      front = 0;
      rear = 0;
   } else {
      rear = (rear + 1) % MAX;  // move circularly
   }
   cqueue[rear] = data;
   printf("Inserted %d\n", data);
}
// Delete (Dequeue)
void dequeue() {
   if (isEmpty()) {
      printf("Queue Underflow! Nothing to delete.\n");
      return;
   }
   printf("Deleted %d\n", cqueue[front]);
   if (front == rear) { // single element
      front = -1;
      rear = -1;
   } else {
      front = (front + 1) % MAX; // move circularly
   }
}
```

```
// Display
void display() {
    if (isEmpty()) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Circular Queue: ");
    int i = front;
    while (1) {
        printf("%d ", cqueue[i]);
        if (i == rear)
            break;
        i = (i + 1) % MAX;  // move circularly
    }
    printf("\n");
}
// Main menu
int main() {
    int choice, data;
    while (1) {
        printf("\n--- Circular Queue Menu ---\n");
        printf("1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                enqueue(data);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
    return 0;
}
```

CIRCULAR QUEUE IMPLEMENTATION USING LINKED LIST

```
#include <stdio.h>
#include <stdlib.h>
// Node structure
struct Node {
    int data;
    struct Node* next;
};
struct Node *front = NULL, *rear = NULL;
// Enqueue (Insert at rear)
void enqueue(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if (front == NULL) {
        front = rear = newNode;
        rear->next = front; // circular link
    } else {
        rear->next = newNode;
        rear = newNode;
        rear->next = front; // maintain circularity
    }
    printf("Inserted %d\n", data);
}
// Dequeue (Delete from front)
void dequeue() {
    if (front == NULL) {
        printf("Queue Underflow! Nothing to delete.\n");
        return;
    }
    if (front == rear) { // only one node
        printf("Deleted %d\n", front->data);
        free(front);
        front = rear = NULL;
    } else {
        struct Node* temp = front;
        printf("Deleted %d\n", temp->data);
        front = front->next;
        rear->next = front; // maintain circularity
        free(temp);
    }
}
```

```
// Display
void display() {
    if (front == NULL) {
        printf("Queue is empty.\n");
        return;
    }
    struct Node* temp = front;
    printf("Circular Queue: ");
    do {
        printf("%d ", temp->data);
        temp = temp->next;
    } while (temp != front);
    printf("\n");
}
// Main Menu
int main() {
    int choice, data;
    while (1) {
        printf("\n--- Circular Queue (Linked List) Menu ---\n");
        printf("1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                enqueue(data);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
    return 0;
}
```