#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 5

// Structure to represent the circular queue

typedef struct {

int\* array;

int front;

int rear;

int size;

} CircularQueue;

// Function to initialize the circular queue

void initializeQueue(CircularQueue\* queue) {

queue->array = (int\*)malloc(MAX\_SIZE \* sizeof(int));

queue->front = -1;

queue->rear = -1;

queue->size = 0;

}

// Function to check if the queue is empty

int isEmpty(CircularQueue\* queue) {

return queue->size == 0;

}

// Function to check if the queue is full

int isFull(CircularQueue\* queue) {

return queue->size == MAX\_SIZE;

}

// Function to write data to the queue

void writeQueue(CircularQueue\* queue, int data) {

if (isFull(queue)) {

// If the queue is full, overwrite the oldest data

queue->front = (queue->front + 1) % MAX\_SIZE;

} else if (isEmpty(queue)) {

// If the queue is empty, initialize front and rear

queue->front = 0;

queue->rear = 0;

} else {

// Move the rear circularly

queue->rear = (queue->rear + 1) % MAX\_SIZE;

}

// Write data to the rear of the queue

queue->array[queue->rear] = data;

// Update the size of the queue

if (queue->size < MAX\_SIZE) {

queue->size++;

}

printf("Data %d written to the queue.\n", data);

}

// Function to read data from the queue

int readQueue(CircularQueue\* queue) {

int data = -1;

if (!isEmpty(queue)) {

// Read data from the front of the queue

data = queue->array[queue->front];

// Move the front circularly

queue->front = (queue->front + 1) % MAX\_SIZE;

// Update the size of the queue

queue->size--;

printf("Data %d read from the queue.\n", data);

} else {

printf("Queue is empty. Cannot read data.\n");

}

return data;

}

// Function to clear the queue

void clearQueue(CircularQueue\* queue) {

// Reset front, rear, and size to empty the queue

queue->front = -1;

queue->rear = -1;

queue->size = 0;

printf("Queue cleared.\n");

}

// Function to free the memory allocated for the queue

void freeQueue(CircularQueue\* queue) {

free(queue->array);

}

int main() {

CircularQueue queue;

initializeQueue(&queue);

// Writing data to the queue

writeQueue(&queue, 1);

writeQueue(&queue, 2);

writeQueue(&queue, 3);

writeQueue(&queue, 4);

writeQueue(&queue, 5);

// Attempting to write more data to a full queue

writeQueue(&queue, 6);

//writeQueue(&queue, 7);

//writeQueue(&queue, 8);

// Reading data from the queue

readQueue(&queue);

readQueue(&queue);

readQueue(&queue);

readQueue(&queue);

readQueue(&queue);

// Attempting to read from an empty queue

readQueue(&queue);

// Clearing the queue

clearQueue(&queue);

// Freeing the memory allocated for the queue

freeQueue(&queue);

return 0;

}