Institute for Advancing Intelligence, TCG CREST
(TCG Centres for Research and Education in Science and Technology)

Introduction to Computer Programming and Data Structures
Homework 02
Topic: Stack and Queue Operations

Implement both the stack and the queue using arrays and dynamic memory allocation.

1. Create a new C file and name it `stack_queue.c`.
2. Implement a `Stack struct` that represents a stack. The struct should have the following fields:
    (a) `int top`: An integer representing the index of the top element of the stack.
    (b) `int capacity`: An integer representing the maximum number of elements the stack can hold.
    (c) `int *array`: A pointer to an array that holds the elements of the stack.
3. Implement a `Queue struct` that represents a queue. The struct should have the following fields:
    (a) `int front`: An integer representing the index of the front element of the queue.
    (b) `int rear`: An integer representing the index of the rear element of the queue.
    (c) `int capacity`: An integer representing the maximum number of elements the queue can hold.
    (d) `int *array`: A pointer to an array that holds the elements of the queue.
4. Implement the following functions for manipulating the stack:
    (a) `Stack createStack(int capacity)`: Initializes a stack with the given capacity.
    (b) `void push(Stack *stack, int element)`: Pushes an element onto the top of the stack.
    (c) `int pop(Stack *stack)`: Removes and returns the top element of the stack.
    (d) `int peek(Stack *stack)`: Returns the top element of the stack without removing it.
    (e) `int isStackEmpty(Stack *stack)`: Returns 1 if the stack is empty, 0 otherwise.
    (f) `int isStackFull(Stack *stack)`: Returns 1 if the stack is full, 0 otherwise.
5. Implement the following functions for manipulating the queue:

(a) `Queue createQueue(int capacity)`: Initializes a queue with the given capacity.

(b) `void enqueue(Queue *queue, int element)`: Adds an element to the rear of the queue.

(c) `int dequeue(Queue *queue)`: Removes and returns the front element of the queue.

(d) `int getFront(Queue *queue)`: Returns the front element of the queue without removing it.

(e) `int isQueueEmpty(Queue *queue)`: Returns 1 if the queue is empty, 0 otherwise.

(f) `int isQueueFull(Queue *queue)`: Returns 1 if the queue is full, 0 otherwise.

6. You should use dynamic memory allocation to allocate memory for the arrays. Remember to free the memory when you are finished using it.

Test your program with several test cases to ensure that it works correctly. You may also want to include error handling for cases where the capacity of the stack or queue is negative or the array is NULL.

Keep your stack_queue.c file. You may require it for some next assignment or exam.