CMSC 331 - Principles of programming language
Homework - 3

1. Determine the types of the following:
   a) fun double x = double (2 * x);
   b) fun apply g y = g(y) + g(y + 1);
   c) fun squash [] = [] |
      squash ({a,b}::c) = ((a+b):int)::(squash c);
   d) fun zipper f a b = f(a,b);
   e) fun ex1 f [] = [] |
      ex1 f (a::b) = (f a)::(ex1 f b);

2. signature QUEUE = sig
   (* A type for a queue with elements of type 'a *)
   type 'a queue

   (* Creates an empty queue *)
   val empty: unit -> 'a queue

   (* Adds an element to the end of the queue *)
   val enqueue: 'a * 'a queue -> 'a queue

   (* Removes the element from the front of the queue. Raises Empty if the queue is empty. *)
   val dequeue: 'a queue -> 'a queue

   (* Returns the element at the front of the queue without removing it. Raises Empty if the queue is empty. *)
   val peek: 'a queue -> 'a

   (* Checks if the queue is empty *)
   val isEmpty: 'a queue -> bool

   (* Applies a function to each element of the queue, from front to back, and returns a new queue with the results *)
   val map: ('a -> 'b) -> 'a queue -> 'b queue
   end;

Implementation Task:
Define the structure that matches the QUEUE signature.
Implement the queue functionality using a list to store elements.
Ensure your implementation correctly handles empty queue cases.

Test Cases:

After implementing the QUEUE structure, create test cases that cover each of the following scenarios:

- Creating an empty queue and checking if it is indeed empty.
- Enqueueing one element, then multiple elements, and verifying the state of the queue after each operation.
- Dequeuing elements from the queue until it is empty, checking the correctness of each removed element, and ensuring the Empty exception is raised when attempting to dequeue from an empty queue.
- Peeking at the front element of the queue without removing it, and ensuring the Empty exception is raised when peeking an empty queue.
- Testing the isEmpty function on both empty and non-empty queues.
- Applying a map function to a queue, such as doubling numbers in a queue of integers, and verifying the output queue reflects the applied function without altering the original queue's state.

Submit your code including all the test cases.