CMSC132 Practice Questions

Problem 1 Sorting

A. What is a comparison sort?

B. When is a sorting algorithm not a comparison sort? Given an example of a sorting algorithm that isn’t a comparison sort.

C. What is a stable sort?

D. What is an in-place sort?

E. What is an external sort?

F. What is the average case complexity of sorting a random array using
   a. bubble sort
   b. heap sort
   c. quick sort

G. What is the worst case complexity of sorting using
   a. selection/exchange sort
   b. tree sort
   c. heap sort
   d. quick sort

H. Are the typical implementations of these sorts stable?
   a. bubble sort
   b. quick sort
   c. merge sort

I. Are the typical implementations of these sorts performed in place?
   a. selection/exchange sort
   b. tree sort
   c. merge sort
Problem 2 Recursion

For this problem, you need to implement a recursive solution to the nqueens problem. The nqueens problem is the problem of, given an N by N chess board, find all placements of N queens such that no queen can attack any other queen.

You are provided with a method that determines if two queens can attack each other:

```java
static boolean canAttack(int r1, int c1, int r2, int c2) {
    return r1 == r2 || c1 == c2 || r1 + c1 == r2 + c2
            || r1 - c1 == r2 - c2;
}
```

For this problem, you need to define a method nqueens(int n) that will print out all placements of n queens on a n x n board (board positions numbered 0..n-1):

```java
static void nqueens(int n) { ... }
```

For example, invoking nqueens(4) would print:

```
[1, 3, 0, 2]
[2, 0, 3, 1]
```

You should use recursion to implement nqueens (e.g., your nqueens method should invoke an auxiliary function that is recursive).

a) What is the base case for the recursive method(s) you give in part (b)?

b) Provide your implementation of nqueens(int n) and any auxiliary functions you need to define