1. a) (5 points) Give an advantage that an array has over a linked list.

b) (5 points) Give an advantage that a linked list has over an array.

2. a) (10 points) Find reasonable constants $c, n_0$ to show that $2n + n \log_{10}(10n) \in O(\log_{10} n)$. Show your work.

b) (10 points) Consider the following "proof" that $\sum_{i=1}^{n} 3n \in O(n)$

$$\sum_{i=1}^{n} 3n = 3n + 3n + ... + 3n$$

$$= O(n) + O(n) + ... + O(n)$$

Given two functions that are each $O(n)$, their sum is also $O(n)$. Therefore, the claim is true.

What is wrong with this argument?
3. a) (10 points) Prove the following identity. Do not use induction.

\[ \sum_{i=0}^{n} a^i = \frac{1 - a^{n+1}}{1 - a} \quad \forall \ a \neq 1 \]

b) (5 points) Why is the above identity not true when \( a = 1 \)?

4. (10 points) Suppose we would like to write an iterator for a binary tree which iterates over the nodes of the tree. Explain briefly how a stack might be used to do this. It is not necessary to write any code; instead, focus on describing the algorithm. Remember, an iterator must support the next() and hasNext() methods.
5. The following does a merge-sort on an array:

```c
void mergesort(int[] array, int start, int end) {
    if(start == end) return;

    int middle = (start + end) / 2;
    mergesort(array, middle + 1, end);
    mergesort(array, start, middle);
    merge(array, start, middle, end);
}

void merge(int[] array, int start, int middle, int end) {
    //merge two sorted segments of array in O(n) time.
    //where n = end - start
}
```

a) (5 points) Give a recurrence relation that describes the worst-case running time of the algorithm.

b) (10 points) Solve the recurrence to obtain the closed form of the running time.

c) (5 points) Suppose we wanted to sort a linked list instead of an array, and merge() now takes two linked list nodes specifying the segments to be merged. How will this affect the running time?
6. (5 points) Suppose you are given two dictionary implementations, one which uses a binary search tree and one which uses a linked list. Explain why the binary search tree implementation will allow faster searches than the linked list implementation will.

7. (20 points) Write a Java implementation of the Queue interface which is based on a pair of stacks. Make it as efficient as possible. You do not need to implement the stack.

```java
public interface Stack<T> {
    public void push(T item);
    public T pop();
    public T peek();
    public boolean isEmpty();
}

public class ArrayStack<T> implements Stack<T> {...}

public interface Queue<T> {
    public void enqueue(T item);
    public T dequeue();
    public boolean isEmpty();
}
```