List and Array Operations

(1) Consider the following (inefficient, but correct) implementation of a function to find the minimum element of a LinkedList of integers:

```
public static int findMinimum(LinkedList<Integer> l) {
    int tempMin = l.get(0);
    for (int i = 1; i < l.size(); i++){
            int current = l.get(i);
            if (current < tempMin) tempMin = current;
    }

    return tempMin;
}
```

Note: The get method above traverses the nodes of the LinkedList, starting at index 0, until it finds the element at index i, then returns the element.

(a) Give the runtime complexity (asymptotic complexity) of running this function on a list with n elements.

(b) Rewrite the function to run in $O(n)$ time.

(2) Give the runtime complexity of the following operations on data structures, and a brief explanation:

1. Inserting a new element into an ArrayList at an arbitrary index.

2. Inserting a new element as the first element of the LinkedList.

3. Removal of an arbitrary element of an ArrayList.

4. Accessing an arbitrary index of an ArrayList.

5. Counting the number of nodes in a Tree.

6. Computing the depth of a balanced tree.

7. Searching for an element in a tree.

8. Searching for an element in a binary search tree.

9. Reversing the order of words in a string. ("Java is fun" becomes "fun is Java")

10. [Extra credit] Calculating whether a number is prime or not.

11. [Hard] Computing the median of numbers in a linked list.

(3) Write down the asymptotic complexity of each of the following functions:

1. $f(n) = \sqrt{n^3 + n^2 + 10}$

2. $f(n) = n!$

3. $f(n) = 2^n + n^2$

4. $f(n) = n \log n + (\log n)^2$

5. $f(n) = 100000n + \log n$

6. $f(n) = n + 0.0000000012^n$

7. $f(n) = \sum_{k=1}^{K} \log n^k$

8. $f(n) = \log \log n$

9. $f(n) = (n + 10)^4$

10. $f(n) = \sqrt{n} + 10 \log n$.

(4) Give the asymptotic complexity of the following recursive function from lecture notes:

```
 /** = a**n. Precondition: n >= 0 */
static int power(int a, int n) {
if (n == 0) return 1;
if (n%2 == 0) return power(a*a, n/2);
return a * power(a, n-1);
}
```