Quiz 2

1) The method `maxAndMin` returns the maximal and minimal element in parameter `list` – an array is used to return two values. If the array has $N$ elements, what is the big-Oh complexity of the call `maxAndMin(list)`? It is one of $O(N)$, $O(N^2)$, $O(\log N)$ or $O(N \log N)$. Justify your answer briefly (3 points).

```java
public static String[] maxAndMin(String[] list){
    String[] ret = new String[2];
    int maxIndex = 0;
    int minIndex = 0;
    for (int k = 1; k < list.length; k++){
        int leg = list[k].compareTo(list[maxIndex]);
        if (leg > 0) maxIndex = k;
        if (leg < 0) minIndex = k;
    }
    ret[0] = list[minIndex];
    ret[1] = list[maxIndex];
    return ret;
}
```

2) The method `counts` below returns the number of occurrences of `a` in `list`. If the array has $N$ elements, what is the big-Oh complexity of the call `counts(list, s)`? It is one of $O(N)$, $O(N^2)$, $O(\log N)$ or $O(N \log N)$. Justify your answer briefly (3 points).

```java
public static int counts(String[] list, String s){
    int count = 0;
    for (int k = 0; k < list.length; k++){
        if (list[k].equals(s)){
            count++;
        }
    }
    return count;
}
```
3) The method `fastcounts` below returns the number of occurrences of a in list. Assume the list array is sorted so that the call to `Arrays.binarySearch` works (the complexity of `binarySearch` is $O(\log N)$ for an N-element array.) If the array has $N$ elements, what is the big-Oh complexity of the call `fastcounts(list, s)`? It is one of $O(N)$, $O(N^2)$, $O(\log N)$ or $O(N \log N)$. Justify your answer briefly – assume that no word occurs more than 25 times (this means that you don’t have to take the number of occurrences of a word into account when determining the complexity, just the size of the array). (3 points)

```java
public static int fastcounts(String[] list, String s) {
    int count = 0;
    int index = Arrays.binarySearch(list, s);
    int first = index;
    int last = index;
    if (first < 0) {
        return 0;
    }
    // s occurs at least once, find the range of occurrences
    while (0 <= first && list[first].equals(s)) {
        first--;
    }
    while (last < list.length && list[last].equals(s)) {
        last++;
    }
    return last - first + 1;
}
```

4) Using a linked list, what is the big-Oh complexity of searching for a specific value? (1 point)