Introduction to Algorithm Analysis

`reorder()` is a method that sorts two array elements.

```
void reorder(int[] array, int i, int j) {
    if (array[i] > array[j]) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
}
```

Figure 1

Question 1. Suppose an array `a` contains the values {`6, 11, 9, 13`}. List the contents of `a` after the method call `reorder(a, 1, 2)`.

Question 2. Suppose we define an **operation** as an *assignment statement*, *arithmetic operation*, or *comparison*. How many operations does the method execute when `reorder(a, 1, 2)` is called?

Question 3. How many operations does the method execute when `reorder(a, 0, 1)` is called?

Question 4. Suppose an array `b` contains the values {`2, 6, 13, 8, 3`}. How many operations does the method execute when `reorder(b, 3, 4)` is called?

Question 5. How many operations does the method execute when `reorder(b, 1, 2)` is called?

Question 6. Is there an upper bound (i.e. maximum amount) on the number of operations that `reorder()` can execute? Why or why not?

Question 7. Does the number of operations the `reorder()` method executes depend on the size of its input (*i.e.*, the number of elements in the input)? Why or why not?

Question 8. We say that the `reorder()` method executes in *constant* time. Another way to say this is that the method is $\mathcal{O}(1)$. Complete the following sentence:

A method is $\mathcal{O}(1)$ (or executes in constant time) if...

Below is a Java method `normalize()` that maps values that are in the range [*min*..*max*] to the range [0..1]:

```
void normalize(double[] array, double min, double max) {
    for (int i = 0; i < array.length; i++) {
        array[i] = (array[i] - min) / (max - min);
    }
}
```

Figure 2

Question 9. Suppose an array `a` contains the values {`5, 15, 10`} and the method is called with the following method call:

```
normalize(a, 5, 15);
```

What are the contents of the array after this method call?

Question 10. How many operations does the method execute when `normalize(a, 5, 15)` is called?
Note: the initialization of the variable `i` executes before the first iteration of the loop. The iteration and comparison statements occur after each iteration of the loop.

Question 11. Suppose the `normalize()` method is called with an array of length 20 as an argument. How many operations are executed by the method?

Question 12. Suppose the `normalize()` method is called with an array of length n as an argument. How many operations are executed by the method?

Question 13. We say that the `normalize()` method runs in *linear* time. Another way to say this is that the method is $\mathcal{O}(n)$. Complete the following sentence:

A method is $\mathcal{O}(n)$ (or executes in linear time) if...

Question 14. We say that *quadratic* time methods are $\mathcal{O}(n^2)$. Complete the following sentence:

A method is $\mathcal{O}(n^2)$ (or executes in quadratic time) if...

Label each of the following methods either $\mathcal{O}(1)$, $\mathcal{O}(n)$, or $\mathcal{O}(n^2)$.

```
int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

Question 15. The `max()` method is $\mathcal{O}(\quad)$. Justify your answer.

```
int maxElement(int[ ] array) {
    int max = array[0];

    for (int i = 0; i < array.length; i++) {
        if (array[i] > max) {
            max = array[i];
        }//end if
    } //end for

    return max;
}
```

Question 16. The `maxElement()` method is $\mathcal{O}(\quad)$. Justify your answer.

```
int maxSubseqSum(int[ ] array) {
    int max = array[0];

    for (int i = 0; i < array.length; i++) {
        int sum = 0;
        for (int j = i; j < array.length; j++) {
            sum += j;

            if (sum > max) {
                max = sum;
            } //end if
        } //end for
    } //end for

    return max;
}
```

Question 17. The `maxSubseqSum()` method is $\mathcal{O}(\quad)$. Justify your answer.

Question 18. We are using the number of operations a method executes as a measure of its run time. In a few complete sentences, explain why we are using this measure of time rather than a wall-clock measure of time (*i.e.*, minutes, seconds, *etc.*).

Question 19. Why is knowing that a method is $\mathcal{O}(n)$ more valuable than knowing that it takes fifteen seconds to execute on a 2.7GHz i7? In the space below, list the pros and cons for each statement.

- "The method is $\mathcal{O}(n)$."

- "The method took 15s on my i7."

Question 20. Is it possible that there are inputs for which a $\mathcal{O}(1)$ method executes more operations than a $\mathcal{O}(n)$ method that has the same specification? Why or why not?