Task 1: Algorithm Analysis (25%)

You may either type or handwrite (neatly!) your solutions. Upload your solutions electronically (e.g., by scanning) to Gradescope, HW3: Algorithm Analysis.

1. Show that 2^{n+1} is by $O(2^n)$ by finding c and n^0 to satisfy the big-O requirement. Explain why your chosen values work.

   n+1 ~= n

   $2^{n+1} = 2^n * 2^1 = 2^n * 2 = 2(2^n) \sim= 2^n$

   If $2*2^n$ is less then and equal to $c(2^n)$, then c is greater then and equal to 2.

2. Show that 2^{2n} is not $O(2^n)$ by showing that it is not possible to fine c and n0 to satisfy the big-O requirement. Note that $2^{2n}=(2^n)^2$.

$2^{2n} = 2^n * 2^n = (2^n)^2$

Suppose $2^{2n} = O(2^n)$, the constant c will become less than 0.

Since 2^n is unbounded, this means no such c can exist.

3. Fill out the table by giving a big-O characterization (and brief justification) of the running time, in terms of n, of each of the following five loops. Think in terms of the number of loop iterations that will be required. Note that the sum of the arithmetic sequence 1, 2, 3, $\cdots$, k is $k2(1+k)$, $\leftarrow$represents an assignment and `for ... m to n do` has the default increment = 1 with both `m` and `n` inclusive.

Pseudo-code	Big-O Characterization and brief justification
Algorithm Loop1 (n): $s \leftarrow 0$ **for** $i \leftarrow 1$ to n **do** $s \leftarrow s + i$	O(N) Linear Based on a single for loop, spend a constant amount of time processing each piece of input data.
Algorithm Loop2 (n): $p \leftarrow 1$ **for** $i \leftarrow 1$ to $2n$ **do** $p \leftarrow p * i$	O(N) Linear The running time is based on the size of N.
Algorithm Loop3 (n): $p \leftarrow 1$ **for** $i \leftarrow 1$ to n^2 **do** $p \leftarrow p * i$	$O(N^2)$ Quadratic For each time the outer loop executes, the inner loop executes n times.

Algorithm Loop4 (n):
$s \leftarrow 0$
for $i \leftarrow 1$ to $2n$ **do**
for $j \leftarrow 1$ to i **do**
$s \leftarrow s + i$
	$O(N^2)$
	The running time of for loop is O(N).
	$O(N) * O(N) = O(N^2)$

Algorithm Loop5 (n):
$s \leftarrow 0$
for $i \leftarrow 1$ to n^2 **do**
for $j \leftarrow 1$ to i **do**
$s \leftarrow s + i$
	$O(N^4)$
	
	The outer loop is $O(N^2)$ by n^2. The running time of two for loops is $O(N^2)$.
	$O(N^2) * O(N^2) = O(N^4)$.

4. Given an ArrayList of initial size n, give a big-O characterization (and justification) of the running time of the following Java function, in terms of n:

```
public void doubleList(ArrayList myList) {
    int size = myList.size();
    for (int i = 0; i < size; i++) {
      int pos = rand.nextInt(myList.size());//rand is a Random object
      myList.add(pos, i);
    }
}
```

Would your answer change if the fourth line instead read "`int pos = myList.size();`"? If so, what would be the new running time and why?

The running time of this Java function is $O(N^2)$. The running time of for-loop is O(N) and Random method is O(N). $O(N)*O(N) = O(N^2)$. If the fourth line instead "`int pos = myList.size();`" the running time does not change, which is $O(N^2)$, because the fourth line only change the output result and can't change the size of the loop.