Problem Set 1

This problem set is due **Wednesday, February 22** at **11:59PM**.

Solutions should be turned in through the course website. **You must enter your solutions by modifying the solution template (in Python) which is also available on the course website.** The grading for this problem set will be largely automated, so it is important that you follow the specific directions for answering each question.

For multiple-choice questions, your grade will be based only on the correctness of your answer. For all other non-programming questions, full credit will be given only to the correct solution which is described clearly and concisely.

Programming questions will be graded on a collection of test cases. Your grade will be based on the number of test cases for which your algorithm outputs a correct answer within time and space bounds which we will impose for the case. Please do not attempt to trick the grading software or otherwise circumvent the assigned task.

---

1. **(15 points)** Order of Growth

   For each group of functions, sort the functions in increasing order of asymptotic (big-O) complexity. Partition each group into equivalence classes such that $f_i(n)$ and $f_j(n)$ are in the same class if and only if $f_i(n) = \Theta(f_j(n))$. (You do not need to show your work for this problem.)

   (a) **(5 points)** Group A:

   $$\begin{aligned} f_1(n) &= n \log n \\ f_2(n) &= n + 100 \\ f_3(n) &= 10n \\ f_4(n) &= 1.01^n \\ f_5(n) &= \sqrt{n} \cdot (\log n)^3 \end{aligned}$$

   (b) **(5 points)** Group B:

   $$\begin{aligned} f_1(n) &= 2^n \\ f_2(n) &= 2^{2n} \\ f_3(n) &= 2^{n+1} \\ f_4(n) &= 10^n \end{aligned}$$

(c) **(5 points)** Group C:

$$f_1(n) = n^n$$
$$f_2(n) = n!$$
$$f_3(n) = 2^n$$
$$f_4(n) = 10^{10^{100}}$$

Solution Format:

Your answer to this problem should be a list of lists of integers. Each sublist should contain the indices of a set of functions which all have the same rate of growth. The order of the indices within the sublist does not matter. The sublists should be ordered from the slowest-growing functions to the fastest.

Example Question:

$$f_1(n) = n$$
$$f_2(n) = 2n^3$$
$$f_3(n) = n + 5$$
$$f_4(n) = n^2$$
$$f_5(n) = n^3$$

Example Answer:

```
# Note that 4 is in a list by itself
# Note that the order of 1 and 3 (and 5 and 2) does not matter
answer_for_example_for_problem_1 = [[1, 3], [4], [5, 2]]
```

2. **(10 points)** Recurrence Relations

   (a) **(5 points)** What is the asymptotic complexity of an algorithm with runtime given by the recurrence:

   $$T(n) = 4T(n/2) + \log n.$$

   1. $\Theta(n)$
   2. $\Theta(n \log n)$
   3. $\Theta(n^2)$
   4. $\Theta(n^2 \log n)$

   (b) **(5 points)** What is the asymptotic complexity of an algorithm with runtime given by the recurrence:

   $$T(n) = 9T(n/3) + n^2.$$

   1. $\Theta(n \log n)$
   2. $\Theta(n^2)$
   3. $\Theta(n^2 \log n)$
   4. $\Theta(n^3)$

Solution Format:

Your answer to this problem should be a single integer for each part. For example, if you thought the answer to part (a) was 5 and the answer to part (b) was 6, then your answer should look like:

```
answer_for_problem_2_part_a = 5
answer_for_problem_2_part_b = 6
```

3. **(20 points)** 2D Peak Finding

   Consider the following approach for finding a peak in an $(n \times n)$ matrix:

   1. Find a maximum element m in the middle column of the matrix.
      - If the the left neighbor of m is greater than it, discard the center column and the right half of the matrix.
      - Else, if the right neighbor of m is greater than it, discard the center column and the left half of the matrix.
      - Otherwise, stop and return m.
   2. Find a maximum element m' in the middle row of the remaining matrix.
      - If the the upper neighbor of m' is greater than it, discard the center row and the bottom half of the matrix.
      - Else, if the lower neighbor of m' is greater than it, discard the center row and the top half of the matrix.
      - Otherwise, stop and return m'.
   3. Go back to step 1.

   (a) **(5 points)** Let the worst-case running time of this algorithm on an $(n \times n)$ matrix be $T(n)$. State a recurrence for $T(n)$. (You may assume that it takes constant time to discard parts of the matrix.)

   (b) **(5 points)** Solve this recurrence to find the asymptotic behavior of $T(n)$.

   (c) **(10 points)** Prove that this algorithm always finds a peak, or give a small $(n \leq 7)$ square counterexample on which it does not.

   Solution Format:

   Your answers to parts (a) and (b) should be Python strings. For example, you may write:

```
answer_for_problem_3_part_a = 'This is a Python string.'
answer_for_problem_3_part_b = '''
Here is a Python multiline string.
It starts and ends with three quotation marks.
'''
```

   If your answer to part (c) is a proof of correctness, then return a string as above. If it is a counterexample matrix, then write the matrix as a list of lists of integers, **not as a string**.

4. **(30 points)** Programming Exercise: Peak In Circle

   Write a function `find_peak_in_circle` that efficiently finds a peak value in a circle of integers. This function should take a list of integers as an input. Two elements in this list are *adjacent* if they are consecutive elements of the list or if they are the first and last element. A peak is an element of the list which is greater than or equal to both of its adjacent elements - your goal is to find the value of any peak.

   You may assume that the input list is non-empty. However, you may not change the entries of the list, and your function should also accept (immutable) tuples. Here are some example test cases that your function should agree with:

```
# Both 4 and 5 are peaks in the array [1, 2, 5, 3, 4]
find_peak_in_circle([1, 2, 5, 3, 4]) in (4, 5)
# The element 3 is not a peak in [3, 2, 1, 4] because it is adjacent to 4
find_peak_in_circle([3, 2, 1, 4]) == 4
```

   Solution Format:

   You should answer this problem by filling in the body of the function `find_peak_in_circle` in the solution template.