DS 320 Algorithms for Data Science **Homework #1**
Fall 2024 Due: Wednesday 9/18

Homework is due via Gradescope at 11:59pm. Solutions **must be typeset in LaTeX**—a template can be found on the course website. You are encouraged to collaborate with up to three other students, but **you must write up your solutions individually**; what is submitted should represent your own words and thoughts. List your collaborators at the top (even if "none").

Wherever you're asked to provide an algorithm or proof, you're allowed to call subroutines from class and use facts that we've proved together in class. You're not, however, allowed to use some subroutine or fact that hasn't been covered in our material yet.

Question 1: What's the Loopin Variant?

(33 points total) Consider the following pseudocode:

Algorithm 1 TestAlg(A).

Input: $A[1 \dots n]$ is an array of integers, indexed 1 to n.
j = 0
while $j + 1 \leq n/2$ **do**
  $j = j + 1$
  $k = n + 1 - j$
  $temporary = A[j]$
  $A[j] = A[k]$
  $A[k] = temporary$
end while
return A

a. (6 points) Briefly state (in English) what this algorithm does.

b. (21 points) Which of the following statements are **loop invariants**, i.e., statements that are true before and after every iteration of the **while** loop? Simply write "true" or "false." (*Hint:* Read carefully!)

(a) The sub-array $A[1 \dots j]$ contains the original contents of sub-array $A[(n+1-j) \dots n]$ in reverse order.

(b) The sub-array $A[(n+1-j) \dots n]$ contains its original contents in their original order.

(c) The sub-array $A[1 \dots j]$ contains its original contents in their original order.

(d) The sub-array $A[1 \dots j]$ contains its original contents in reverse order.

(e) The sub-array $A[(n+1-j) \dots n]$ contains its original contents in reverse order.

(f) The sub-array $A[(n+1-j) \dots n]$ contains the original contents of sub-array $A[1 \dots j]$ in reverse order.

c. (6 points) What is this algorithm's runtime? That is, what is the *simplest* function $f(n)$ for which you could say this algorithm's runtime is $\Theta(f(n))$? Explain your answer.

Question 2: What is the Second Highest?

(33 points total) You run into a friend you haven't seen for a while. It turns out you both are using pedometers to count the number of steps you make every day. Your friend works in a warehouse and walks a lot. He decides to flex on you by telling you his highest number of steps in a single day. At first you think that you can't beat him, but then you recall that day when MBTA shut down the green line and, in desperation, you decided to just walk everywhere. You pull out your pedometer and it turns out that your number of steps that day is your personal best and it is exactly the same!

You then decide to compare your second highest numbers but since your pedometer doesn't directly give you this number, you have to figure it out after getting back home. Your task is to give an algorithm that *efficiently* solves this problem. It takes in an array $A[1 \dots n]$ of numbers, which is the data you copied from your pedometer, and returns the second highest number in the array. If the highest value is repeated, then your algorithm should return that value. Use only comparison operations ($<, >, \leq, \geq$ or $=$) and assignments (copying/storing). You can assume that $n \geq 2$.

As with all algorithms problems, you should:

a. (13 points) Give clear pseudocode for your algorithm. Spell out any subroutines that your algorithm calls. You will be graded on the pseudocode's clarity, readability, and simplicity. As with all writing, this is an exercise in clear communication. Your audience is other students in this class. Any one of them should be able to take your pseudocode and turn it into working code in any high–level language.

b. (15 points) Argue formally that your algorithm is correct on all input instances.

Hint: Find a sub-claim to prove, like a loop invariant or induction on a property that holds as i increases.

c. (5 points) Give as tight as possible analysis of your algorithm's running time up to a constant, i.e., express it using the big–O notation and state it using a function as simple as possible.

Question 3: The O–Omega–Theta Fraternity

(34 points total) You might find the following reminders useful for the following questions:

By definition, $b^{\log_b(x)} = x$.	$\log_b(xy) = \log_b x + \log_b y$.
$\log_b(x/y) = \log_b x - \log_b y$.	$\log_b(x^n) = n \log_b x$.
$\log_b x = \log_a x / \log_a b$.	$\binom{n}{k} = \frac{n!}{k!(n-k)!}$

a. (20 points) Sort the following functions from smallest to largest *set* of big–O asymptotic order of growth, noting if any are of the *same* asymptotic order. No explanation is necessary.

Recall that $O(g(n))$ is the *set* of functions for which there exist constants c_1 and c_2 such that for all $n \geq c_1$, $f(n) \leq c_2 g(n)$. For example, $O(n^3) \subseteq O(n^4)$ since any function upper–bounded by $c_2 n^3$ is also upper–bounded by $c_2 n^4$.

- $O(n \log n)$
- $O(\sqrt{n})$
- $O(\log n)$
- $O(n^2)$
- $O\left(n^{3/4}\right)$
- $O(2^n)$
- $O(n)$
- $O(n!)$
- $O(n^{1{,}000{,}000})$
- $O(n^{1/\log n})$
- $O(\log(n!))$
- $O((3/4)^n)$
- $O\left(\binom{n}{2}\right)$

b. (13 points) For the following two pairs of functions, labeled f and g, determine whether $f(n)$ is big–O, Omega, and Theta of $g(n)$. If so, for the first two, give constants c_1 and c_2 that prove it, that is, that satisfy the definition: such that for all $n \geq c_1$, $f(n) \leq c_2 g(n)$ (for $O(g(n))$) or $f(n) \geq c_2 g(n)$ (for $\Omega(g(n))$).

(a) $f(n) = \log n, \quad g(n) = \log\left(n^2\right)$

(b) $f(n) = n, \quad g(n) = (\log_2 n)^4$

(c) $f(n) = (n+1)!, \quad g(n) = (n-1)!$

You might want to fill out the following table:

$f(n)$	$g(n)$	$f(n) = O(g(n))$? $(c_1 = , c_2 =)$	$f(n) = \Omega(g(n))$? $(c_1 = , c_2 =)$	$f(n) = \Theta(g(n))$?
$\log n$	$\log\left(n^2\right)$			
n	$(\log_2 n)^4$			
$(n+1)!$	$(n-1)!$			