Please enter your name and uID below.

Name: Devin White

uID: U0775759

Collaborators, if any, and how you collaborated:

Submission notes

- **(PS0 specific)** Significant points be rewarded for answers that give a clear legitimate effort. The goal of this assignment is to learn the course expectations for written proofs.
- **(PS0 specific)** Due at 11:59 pm (midnight) on Thursday, Sep 1, 2021. *No late submissions will be accepted.*
- Solutions must be typeset using one of the template files. For each problem, your answer must fit in the space provided (e.g. not spill onto the next page) *without* space-saving tricks like font/margin/line spacing changes.
- Upload a PDF version of your completed problem set to Gradescope.
- Teaching staff reserve the right to request original source/tex files during the grading process, so please retain these until an assignment has been returned.
- Please remember that for problem sets, collaboration with other students must be limited to a high-level discussion of solution strategies. If you do collaborate with other students in this way, you must identify the students and describe the nature of the collaboration. You are not allowed to create a group solution, and all work that you hand in must be written in your own words. Do not base your solution on any other written solution, regardless of the source.

1. (Total Induction)

   (a) **Find the polynomial**

   - $1 + \sum_{k=5n}^{7n} k = 1 + (5n + (5n + 1) + (5n + 2) + ...7n$
   - **if** $n = 1 \rightarrow 1 + (5(1) + 6 + 7) = 19$ **if** $n = 2 \rightarrow 1 + (5(2) + 11 + 12 + 13 + 7(2)) = 61$ **if** $n = 3 \rightarrow 1 + (5(3) + 16 + 17 + 18 + 19 + 20 + 7(3)) = 127$
   - **It can be seen that the algorithm runs approximately** $(2n + 1)$ **times**
   - **if** n=1, (x(2n+1))+1 = 19, x=6 **if** n=2 (x(2n+1))+1 = 61, x=12, **if** n=3 (x)(2n+1))+1 = 127, x = 18
   - The result appears to be 6(n) multiplied by the amount of iterations: guess for polynomial is 6n(2n+1)
   - Checking using summation rules: $1+\sum_{k=1}^{7n} k-\sum_{k=1}^{5n-1} k \rightarrow 1+\frac{7n(7n+1))}{2}-\frac{(5n+1)(5n+1-1))}{2} = 1 + \frac{(49n^2+7n)-(25n^2-5n))}{2} = 1 + \frac{(24n^2+12n)}{2} = 12n^2 + 6n + 1$
   - **The Polynomial is:** $1 + (6n(2n + 1))$ or $12n^2 + 6n + 1$

   (b) **Prove by induction**

   - **Base Case:** $n = 1$
     i. $1 + \sum_{k=5(1)}^{7(1)} k \rightarrow 1 + (5(1) + (5(1) + 1) + 7(1)) = 19$
     ii. $12(1)^2 + 6(1) + 1 = 19$

     It is shown that $1 + (5n + (5n + 1) + (5n + 2) + ... + 7n = 12n^2 + 6n + 1$ for n = 1. So the base case is proven
   - **Inductive Step:**
     - **Assume True for m=n:** $1 + \sum_{k=5(1)}^{7(1)} k \rightarrow 1 + \frac{7n(7n+1))}{2} - \frac{(5n+1)(5n+1-1))}{2} \rightarrow 12n^2 + 6n + 1$
     - **Show for m=n+1**: $1+\sum_{k=5(n+1)}^{7(n+1)} k \rightarrow 1+\frac{7(n+1)(7(n+1)+1))}{2}-\frac{(5(n+1)+1)(5(n+1)+1-1))}{2} = 1+\frac{(7n+7)(7n+8)}{2}-\frac{(5n+4)(5n+5)}{2} = 1+\frac{49n^2+56n+49n+56}{2}-\frac{25n^2+25n+20n+20}{2} = \frac{49n^2+56n+49n+56-25n^2+25n+20n+20}{2} = 1 + \frac{24n^2+60n+36}{2} = 12n^2 + 30n + 19$
       Or plugging in to the original polynomial: $12(n + 1)^2 + 6(n + 1) + 1 = 12n^2 + 30n + 19$
     - If $m = 1$ then $n = 2$: $= (12(1)^2 + 30(1)) + 19 = (12 + 30 + 19) = 61$ Same as n=2 shown above
     - if $m = 2$ then $n = 3 = (12(2)^2 + 6(2)) + 19 = (48 + 60 + 19) = 127$ Same as n=3 shown above
     - **Thus the polynomial is proven for all n**

   (c) **Show that the polynomial is** $O(n^2)$

   - **Definition:** $f(n)$ is $O(g(n))$ if $f(n) \leq Cg(n)$ for $n \geq k$ **Or In other words, f = O(g) IF f grows no faster than g, for a constant** $c > 0$ **such that** $f(n) \leq C * g(n)$
   - **Assume that:** $f(n) = 12n^2 + 6n + 1$**,** $g(n) = n^2$**, and** $k = 1$**.**
     Therefore: $12n^2 + 6n + 1$ is $O(n^2)$ if $12n^2 + 6n + 1 \leq Cn^2$ for $n \geq 1$
     if $n \geq 1$ then $12(1)^2 + 6(1) + 1 \leq C(1)^2$
     Therefore: $12 + 6 + 1 \leq C \rightarrow 19 \leq C$
     And $C \geq 19$, and $n \geq 1$
   - Thus $12n^2 + 6n + 1 \leq C * n^2$ for $C \geq 19$ and $n \geq 1$ Therefore by the definition of big O stated above: $12n^2 + 6n + 1$ is $O(n^2)$
   - **Thus it is proven that** $12n^2 + 6n + 1$ **is** $O(n^2)$

2. (Vaccination)

- **Statement to prove:** The stated algorithm ensures that all in-hospital workers are vaccinated before all at-home workers based upon the provided algorithm.
- **Prove by contradiction:**
  (a) **Known facts:**
    - n equals the number of employees at the company
    - Location and ranks consists of arrays of size n(number of employees)
    - Lines **2-12** is using bubblesort to sort employees by location and rank
      * *(Bubble sort works by repeatedly swapping adjacent elements until they are sorted correctly)*
    - location takes precedence first**(lines 4-6)**, then rank if locations match**(lines 7-10)**

  (b) **Assumption:** There exists an at-home worker who is vaccinated before any in-Hospital worker in the provided algorithm.

  (c) Using the above fact that the algorithm is using bubblesort, we can see that locations and ranks are sorted thus:
    - **on line 4:** if the next array element(j+1) is in-hospital, and the current array element(j) is at home, then the rank of array[j+1] is swapped with the rank of array[j] **(line 5)** and location of array[j+1] is swapped with the location of array[j]**(line 6)**
      * *(In other words, J+1 is now at J. If J+1 was originally in-hospital and J at-home)*

    - **on line 7:** if the locations of array[j] and array[j+1] are the same:
      if the rank of array[j] is less than array[j+1](lower ranking)**(line 8)** then the ranks and locations of array[j] and array[j+1] are swapped**(lines 9 and 10 respectively)**

    - Based on our Earlier assumption that there exists an at-home worker who is vaccinated before any in hospital worker by the algorithm, we can already see there is a contradiction based on the algorithms behavior mentioned above.
      * if j is at-home, and j+1 is at-hospital, then the locations and ranks of j+1 and j are swapped making j+1 now earlier in the array(now at j), as seen on line 4 of the algorithm.
      * This will happen always when j is at-home and j+1 is in-hospital, as the if statement on line 4 is always checked, Meaning there is no situation in which an at-home worker is vaccinated before an in-hospital worker.
      * if j+1 is at home, and j is at-hospital, then the locations and ranks of j+1 and j are completely unchanged.
      * This will happen in any case where j+1 is at home and j is at the hospital
      * Since our earlier assumption that there exists some at-home worker whom is vaccinated before any in-hospital worker is always false, then it stands to reason that no at-home worker is vaccinated before any-in hospital worker

      Thus the original statement is proven!

3. (CattleSorting)

When referring to lines or pens, obviously this would be equal to some "holding" data structure. E.G an array
First cow would mean index 0 of that data structure

(a) create k lines or pens of cows, where k is equal to the total number of breeds of cow, we will call these lines 1 through k.

(b) go through the line of cows of size n, 1 at a time(for loop)

(c) starting with the first cow in the original line, pull it out and put it in line 1

(d) move to the next cow in the original line, compare it to the breed of the very first cow in line 1. if they match, put it in that line at the end of the line.

(e) if the cow does not match the first cow in that line compare it to the first cow in the next line.

(f) if the next cow line is empty, put that cow as the original founding member of that line of cows She is now Queen!! First of her name! (or she is index 0 if you want to be pendantic).

(g) repeat lines d-f until all cows from the original cow line are counted/accounted for.

(h) When all cows are put into their proper lines, make a new big line

(i) take each line/pen of now sorted cows, and combine them, one at a time, into a new gloriously large line of cows. for example, all the cows from line 1, then all the cows from line 2, ... all the way to line k.

(j) Congratulations! All your cows are now properly sorted, and your neighbor is absolutely green with envy over your beautiful, perfectly sorted livestock!

This algorithm is O(nk) run time because for each breed of cow(where the number of breeds is K) you are creating some separate structure to hold those cows.Essentially, you are going through the total list of cows a single n time, during this time, you are doing K comparisons, comparing each individual cow in the array to the first cow in each of these k lines/pens. This can be thought of as doing n*k total comparisons, which is an algorithmic run time of O(nk)