I certify that every answer in this assignment is the result of my own work; that Ihave neither copied off the Internet nor from any one else's work; and I have notshared my answers or attempts at answers with anyone else.

1: (a) For each of the following, first answer Yes/No, and then prove your answer.
i. Is I2 valid on initialization,
ii. Is I2preserved by each iteration, and iii. Is I2 useful at termination (i.e., it gets used in part (ii) for I1).
(b) For each of the following, first answer Yes/No, and then prove your answer.
i. Is I1 valid on initialization (i.e., when pass has just been assigned 1)?
ii. Is I1 preserved by each iteration? (You may make use of I2.) and
iii. Is I1 appropriate (i.e., does it demonstrate that the algorithm has achieved its intent) at termination ?
iv. If your answer any of the above, then make the smallest possible correction and justify your modified invariant.

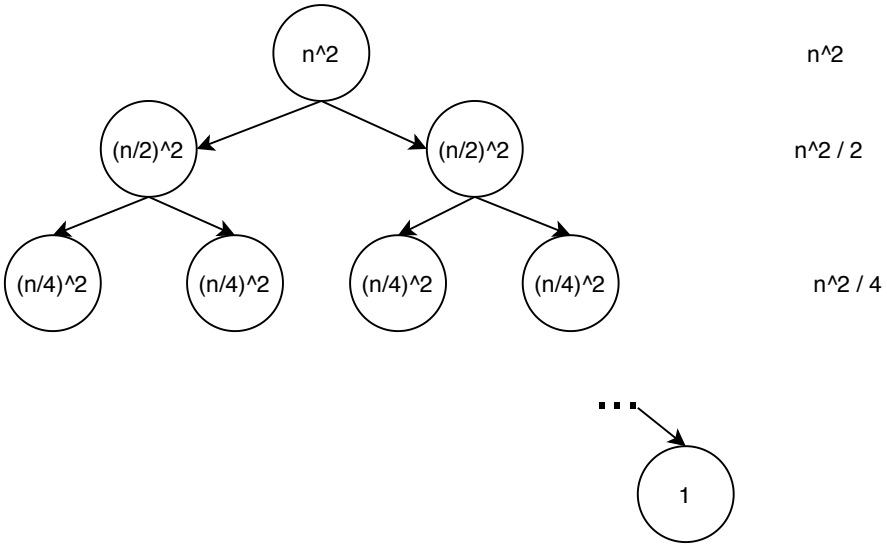
(a)
i: Yes because none of the values have been touched, all of the conditions of I_2 should still hold true for any input
ii: Yes, I_2 is preserved by each iteration. Where ever J is referring to in the array, it will be sorted for every element previous to it so J is greater than or equal to the previous elements. No element are added at this point so any elements before element j would have been some permutation of the previous iteration and anything after it would not have been touched yet so it is unchanged
iii: Yes, I_2 is useful at termination because it meets all its requirements to be used by I_1. i.e. The last element in which J is pointing to is greater than every element previous to it, and no element are added so the elements previous are a sorted permunation of the previous iteration and there is nothing changed about anything after the jth element
(b)
i. No, because you cannot prove that all values in that segment are greater than or equal to the values in A[1 ... (n-pass+ 1)] because nothing has be referenced or can be proven upon initialization
ii. Yes I_1 is preserved by each iteration because with every other iteration we can prove the last element to be the greatest in the array, and no elements are added so it meets all of the conditions for I_1
iii. No I_1 is not appropriate at termination because $A[(n - pass + 2) \dots n]$ does not guarantee that the first two elements are sorted, just that they are less than any of the other elements in the array
iv. Change $A[(n - pass + 2) \dots n]$ to $A[(n - pass + 1) \dots n]$

2: Using the Substitution Method, verify the solution T(n) ==$\Omega(n^2)$for the following recurrence equations:
$T(1) = 1$
$T(n) = n + T(n-1)$ **for n > 1**

Lets assume that T(n) = $\Omega(n^2)$ and therefore
$T(n) \leq cn^2$
$T(n) \leq c(n-1)^2 + n$

$T(n) \leq c(n^2 - 2n + 1) + n$
$T(n) \leq cn^2 + n(1 - 2c) + c$ which holds true for c > 0.5
Therefore T(n) == $\Omega(n^2)$

3: Given the following recurrence equations, find a solution for T(n)using a recursion tree. (You can assume that n is a power of 2 in order to eliminate floors and ceilings.)You must derive a polynomial in n and then infer a Θ()bound without deriving the constants c_1,c_2 and the threshold n_0.
T(1) = 1
T(n) = $n^2 + 2T(n/2)$**for n>1**



$T(n) = n^2 + \frac{n^2}{2} + \frac{n^2}{4} \cdots + \frac{n^2}{2^n} = \sum_{i=0}^{n} \frac{i^2}{2^i} = \Theta(n^2)$ Within each node is the COST of that node and to the right is the cost of that level within the tree.
For instance the children of the root should be $n/2$ and the cost of that node is the value squared because of the n^2 within the equation $T(n) = n^2 + 2T(n/2)$