**Algorithmics Unit 4 Week 1 – Submission Task**

1. Evaluate:

5+9+13=33 [if n=i]

Or 4 x (4n+1) = 12n+3 [if n is independent of i]

6+12+20+30 [if n=i]

Or 5 x n(n+1) = 5n^2+5n

1. If find in terms of :

= 4k

= k+50

1. Prove by induction (for all
2. Consider the following algorithm, which searches for ‘x’ in array ‘A’ and returns either the index (if A contains x) or -1 if A does not contain x:

def search(A,x):

n = length of array [initialisation invariant]

for i = 1 to n:

if A[i] = x then return I [maintenance invariant]

return -1 [termination invariant]

(Here indices start at 1, and A[2:4] includes A[2],A[3] and A[4].)

1. Which of the following are loop invariants?
2. At the start of each iteration, if A contains x then the subarray A[i:n] contains x.
3. At the start of each iteration, if A contains x then the subarray A[i+1:n] contains x.
4. At the start of each iteration, A[1:i] does not contain x.
5. At the start of each iteration, A[1:i-1] does not contain x.
6. At the start of each iteration, if A[i:n] does not contain x then x is not in A.
7. At the start of each iteration, if A[i+1:n] does not contain x then x is not in A.
8. At the start of each iteration, if A[i-1:n] does not contain x then x is not in A.

A. "At the start of each iteration, if A contains x then the subarray A[i:n] contains x." This is valid because if x is in A and hasn't been found yet, it must be in the remaining unexamined portion of the array. At the start of each iteration, A[i:n] represents the part of the array that hasn't been checked yet.

C. "At the start of each iteration, A[1:i] does not contain x." This is valid because the loop has checked all elements from index 1 up to (but not including) i, and hasn't found x. If it had found x, it would have returned immediately.

D. "At the start of each iteration, A[1:i-1] does not contain x." This is valid for the same reason as C, but it's a slightly weaker statement. It only claims that x is not in the subarray up to i-1, whereas C includes one more element (up to i).

E. "At the start of each iteration, if A[i:n] does not contain x then x is not in A." This is valid because A[i:n] represents the part of the array that hasn't been checked yet. If this remaining part doesn't contain x, and we know from C that the part we've already checked (A[1:i]) doesn't contain x, then x is not in A at all.

(b) For the first loop invariant you have identified, explain whether or not it remains true after the loop has completed.

1. If the loop found x and returned an index: In this case, the loop terminated early by returning the index where x was found. The invariant was true up until x was found, which caused the function to return immediately.
2. If the loop completed without finding x: If the loop completed without finding x, we know that x is not in A at all. So the premise of the invariant ("if A contains x") is false, which makes the entire statement vacuously true.

Therefore, after the loop has completed:

* If x was found, the invariant doesn't apply because the loop terminated early.
* If x was not found, the invariant is vacuously true because its premise is false.