Revision and CA1 Preparation

1. What is the time-complexity of the following piece of code when using the Big-O notation?

sum(int n) {

int total = 1;

for (int i = 1; i <= n; i++) {

total += i;

}

return total;

}

1. O(n2)
2. O(n)
3. O(log n)
4. O(n log n)
5. What is the complexity as Big-O Notation for the following polynomial expression:

f(n) = 5n3 + 7n2 + 3n + 1

1. O(5n3)
2. O(n3)
3. O(n2)
4. O(n)
5. Consider that you have a collection of numbers represented as {7,1,2,9,4} that has to be sorted in descendent order using the **Insertion** sort algorithm. How the collection looks after the second iteration of the algorithm?
6. { 7, 2, 1, 9, 4 }
7. { 1, 2, 7, 9, 4 }
8. { 7, 9, 4, 1, 2 }
9. { 1, 7, 2, 9, 4 }
10. Write the **CountMovedElements\_InsertionSort()** method that
    1. sorts the elements in ascendant order (based on Insertion sort algorithm) and
    2. counts and returns how many elements WERE NOT removed from their initial location since they were in the correct location. The method should return at the end the counted number.

*Exemplification*

*Initial list:* { 1, 7 2, 4, 9 }

*After insertion sorting:* { 1, 2 4, 7, 9 }

*Solution:* 2 elements ( 1 and 9) were kept in their original position

Note: use the skeleton\_Exercise4 project provided on Moodle