MOE 1.5 Asymptotic Complexity Exercise

H. Diana McSpadden (hdm5s)

1. ►
```
def measure(inputList):
int n = len(inputList)
int sum = 0;
for i in range(0, n):
        for j in range(0, 5):
                sum += j * inputList[i]
        for k in range(0, n):
                sum -= inputList[k]
```

The asymptotic complexity of this algorithm is: $O(n^2)$, because the i loop and k loops are both range(0,n). The j loop is ignored for two reasons: 1) the range is 0 to 5 so it runs in constant time, and also because even if the loop was range 0 to n, Bog O notation describes how the algorithm grows which would still only be by n^2.

2. ►
```
def addElement(ele):
        myList =[]
        myList.append(666)
        print myList
```

The asymptotic complexity of this algorithm is: $O(1)$

3. ►
```
num = 10

def addOnesToTestList(num):
        testList = []
        for i in range(0, num):
                testList.append(1)
                print(testList)

        return testList
```

The asymptotic complexity of this algorithm is: $O(n)$ where n is num.

4. ►
```
testList = [1, 43, 31, 21, 6, 96, 48, 13, 25, 5]

def someMethod(testList):
        for i in range(len(testList)):
                for j in range(i+1, len(testList)):
                        if testList[j] < testList[i]:
                                testList[j], testList[i] = testList[i], testList[j]
                        print(testList)
```

The asymptotic complexity of this algorithm is: O(n * (n-1)) == O(n^2 - n), but we ignore lower order elements so the complexity is: **$O(n^2)$.**

5. ►
```
def searchTarget(target_word):
        for (i in range1):
                for (j in range2):
                        for (k in range 3):
                                if (aList[k] == target_word):
                                        return 1
                return -1
        return -1
```

The asymptotic complexity of this algorithm is: $\boldsymbol{O(i * j * k)}$ where i is range 1, j is range2, and k is range 3.

6.
```
def someSearch(sortedList, target):
        left = 0
        right = len(sortedList) - 1
        while (left <= right):
                mid = (left + right)/2
                if (sortedList(mid) == target):
                        return mid
                elif(sortedList(mid) < target):
                        left = mid + 1
                else:
                right = mid - 1
        return -1
```

The asymptotic complexity of this algorithm is: $\boldsymbol{O(\log_2 n)}$