Name: Jacob Yanicak

ID Number (Last 4 digits ONLY): 7528

Observe the Collaboration Policy of the course (Handout 1). Same guidelines as HW1.

Problem 1. (9pts)
(a) Use the master method to solve the following recurrences. Just the answer.

$(1) T(n) = 16T(n/4) + n^2 \lg n.$ T(n) = Θ((nlgn)^2)

$(2) T(n) = 16T(n/4) + n \lg n.$ T(n) = Θ(n^2)

$(3) T(n) = 4T(n/4) + n^2 \lg n.$ T(n) = Θ(n^2*lgn)

(b) Solve exactly the following recurrence by providing the answer : $T(n) = 2T(n/2) + 4n$, where $T(8) = 8$.

T(n) = 4nlgn - 11n

(c) Use the substitution method to solve the following recurrence. Just the answer plus constants. Give as tight an answer as you can. You may assume that n is a power of 2. $T(n) = T(n/2) + n$, with $T(2) = 5$.

n <= T(n) <= 5n c1 = 1, c2 = 2

Problem 2. (6pts)
(Decision Problem: YES or NO answer. Algorithm universe what has been done in class.)
You are given n keys in array $A[0..n-1]$ in arbitrary order. We want to determine if they are distinct If this is the case print YES. Otherwise, if at least one key appears twice or more print NO.
(a) Give a space and time efficient algorithm that solves this problem (8 lines max). Give its running time and space requirements using tight asymptotic notation.
(b) Give a time efficient algorithm that solves this problem (8 lines max). Give its running time using tight asymptotic notation.

a)
```
A = InsertionSort(A)
print(Check(A))

function Check():
   for (int i = 0 to A.length - 2):
      if A[i] == A[i+1]:
         return "NO"
   return "YES"
```

Time Complexity = O(n^2)
Space Complexity = Θ(1)

Using Insertion Sort algorithm to sort the array in-place first, then checks for duplicates by iterating through the sorted array.

b)
```
A = MergeSort(A)
print(Check(A))

function Check():
   for (int i = 0 to A.length - 2):
      if A[i] == A[i+1]:
         return "NO"
   return "YES"
```

Time Complexity = O(nlogn)

Using Merge Sort algorithm to sort the array first, then checks for duplicates by iterating through the sorted array.