– I certify that every answer in this assignment is the result of my own work; that I have neither copied off the Internet nor from any one else's work; and I have not shared my answers or attempts at answers with anyone else.

1: For each of the following, indicate whether the statement is TRUE or FALSE. You must also explain the reason for your answer. Do not write pseudo-code. Provide answers that are brief but rich in content.

T / F INSERTIONSORT is a stable sorting algorithm

This statement it true because INSERTIONSORT will keep the original order of elements that have the same key.

2: Assume that the HEAPIFY algorithm we have seen in class is coded as procedures and executed on a normal computer where parameters are pushed on a stack on a recursive call and popped upon return.
Suppose each such recursive call requires $\Theta(1)$ stack space.
The maximum amount of such stack space used at any time during a computation is defined to be the stack depth.
Write recurrence relations for C(n), where the input to HEAPIFY is an array of size n.

This is the code that we recieved in class:

```
HEAPIFY(A, i) /* MAX-HEAPIFY */
1   l ← left(i)
2   r ← right(i)
3   if l ≤ A.heapsize and A[l] > A[i]
4       then largest ← l
5       else largest ← i
6   if r ≤ A.heapsize and A[r] > A[largest]
7       then largest ← r
8   if largest <> i then
9       exchange A[i] with A[largest]
10      HEAPIFY(A, largest)
```

Also given in the slides we can find:
C(0) = c'
C(n) = c + C(n-1)
We can use the bottom up method to conclude:
C(1) = c + C(0) = c + c'
C(2) = c + C(1) = c + c + c'
C(3) = c + C(2) = c + c + c'
...
C(n) = nc + c' = O(lgn)

3: The operation HEAPINCREASEKEY(A,i,δ)increases the value of node number i of (max) heap A by amount δ, which could be either positive or negative. Of course, the heap property is re-established efficiently. Write an algorithm for HEAP-INCREASEKEY(using the notation and model we have used in class) that runs in O(lgn) time for an n-element heap.
It is crucial that you explain your algorithm clearly using comments.the grader cannot understand your algorithm, he may presume that it is wrong. Explain (no need to prove) why your algorithm attains the O() bound.

HEAPINCREASEKEY(A, i, δ)
A[i] = A[i] + δ
if $\delta < 0$ do
  HEAPIFY(A,i)
| > calling heapify will insure that the children of this nod is checked to be in the correct place
else do
  while parent(i) < A[i] do
    exchange parent(i) with A[i])
    HEAPIFY(A, i)
| > if the parent of the node is less than the node i, this should swap the values and make sure
| >that the subtree at that node is still a heap.
| > it should keep doing this process until parent node is not less than node i