Test-2 (20 points each question, Total=100pts)

Name:_____

Question 1.

Consider the following algorithm to check connectivity of a graph defined by its adjacency matrix.

**Algorithm** *Connected*$(A[0..n-1, 0..n-1])$

```
//Input: Adjacency matrix A[0..n-1, 0..n-1]) of an undirected graph G
//Output: 1 (true) if G is connected and 0 (false) if it is not
if n = 1 return 1       //one-vertex graph is connected by definition
else
    if not Connected(A[0..n-2, 0..n-2]) return 0

        else for j ← 0 to n-2 do
                 if A[n-1, j] return 1
             return 0
```

Does this algorithm work correctly for every undirected graph with $n > 0$ vertices? If you answer "yes," indicate the algorithm's efficiency class in the worst case; if you answer "no," explain why.

Question 2.

Shellsort (more accurately Shell's sort) is an important sorting algorithm which works by applying insertion sort to each of several interleaving sublists of a given list. On each pass through the list, the sublists in question are formed by stepping through the list with an increment $h_i$ taken from some predefined decreasing sequence of step sizes, $h_1 > ... > h_i > ... > 1$, which must end with 1. (The algorithm works for any such sequence, though some sequences are known to yield a better efficiency than others. For example, the sequence 1, 4, 13, 40, 121, ... , used, of course, in reverse, is known to be among the best for this purpose.)

a. Apply shellsort to the list

$$S,\ H,\ E,\ L,\ L,\ S,\ O,\ R,\ T,\ I,\ S,\ U,\ S,\ E,\ F,\ U,\ L$$

b. Is shellsort a stable sorting algorithm?

Question 3.

a. Design a version of binary search that uses only two-way comparisons such as $\leq$ and $=$. Implement your algorithm in the language of your choice and carefully debug it: such programs are notorious for being prone to bugs.

b. Analyze the time efficiency of the two-way comparison version designed in part a.

Question 4.

a. Solve the recurrence relation for the number of key comparisons made by mergesort in the worst case. (You may assume that $n = 2^k$.)

b. Set up a recurrence relation for the number of key comparisons made by mergesort on best-case inputs and solve it for $n = 2^k$.

c. Set up a recurrence relation for the number of key moves made by the version of mergesort given in Section 5.1. Does taking the number of key moves into account change the algorithm's efficiency class?

Question 5.

a. Design an efficient algorithm for finding and deleting an element of the smallest value in a heap and determine its time efficiency.

b. Design an efficient algorithm for finding and deleting an element of a given value $v$ in a heap $H$ and determine its time efficiency