1. **(5%) Multiple Choice.** Consider the algorithm *Multiply(X, n)* below. *X* is an array storing *n* integer values. For the best and worst cases, what are the asymptotic time complexities of the following algorithm, respectively? Please provide brief explanation. (answer 2%, explanation 3%)

```
Algorithm Multiply(X, n)
  answer = 0; i = 0
  while i ≤ n/4 do
      j = 0; i = i + 1
      while X[j] < 0 and j < n/8 do
          answer = answer + X[i] * X[j]
          j = j + 2
  return answer
```

(a) Best case O(1), worst case O(n).
(b) Best case O(log n), worst case O(n).
(c) Best case O(n), worst case O(n^2).
(d) Best case O(1), worst case O(n^2).
(e) Best case O(n), worst case O(n).

2. **(5%) Multiple Choice.** Suppose *T* is a full binary tree. Every node *v* of *T* has a variable *var*, which can be accessed with *v.var*. According to the algorithm below, if we call on the root of *T*, what is the meaning of returned value? Here *v.left* and *v.right* are the left and the right children of *v*, respectively. Please provide brief explanation. (answer 2%, explanation 3%)

```
Algorithm Tree(v)
  if v is leaf
      v.var = 0
      return 0
  else
      leftnum = Tree(v.left)
      rightnum = Tree(v.right)
      v.var = 1 + max{v.left.var, v.right.var}
      if v.left.var < v.right.var
          return leftnum + rightnum + 1
      else
          return leftnum + rightnum
```

(a) The number of nodes *v* in the tree such that the left child of *v* is at a smaller depth than the right child of *v*.

(b) The number of nodes *v* in the tree such that the left subtree of *v* has a smaller height than the right subtree of *v.*

(c) The number of nodes *v* in the tree such that the left subtree of *v* has less nodes than the right subtree of *v*.

(d) The number of nodes *v* in the tree such that the left subtree of *v* has less internal nodes than the right subtree of *v*.

(e) The number of nodes *v* in the tree such that the left subtree of *v* has less leaf nodes than the right subtree of *v*.

3. **(8%) Please answer the following questions about trees.**

(a) **(4%)** Please draw how an initially-empty binary search tree would look like after the following numbers are inserted in the given order: {15, 3, 2, 18, 6, 25, 17, 22}. No need to show it in a step-by-step fashion; you only need to draw the final result (a binary search tree, called *T*).

(b) **(4%)** Show the resulting binary search tree after deleting the number 18 from *T* (*T* is the BST constructed in (a)). Please draw the **<u>two possible outcomes</u>**. You only need to show the resulting tree, and briefly explain your steps.

4. **(7%)** In an arbitrary binary search tree of *n* values, we'd like to perform a lookup to search an element in the tree that matches a given key value.

(a) **(4%, each 2%)** What is its worst-case time complexity (in big-O notation)? Please give an example with brief explanation.

(b) **(3 %, each 1%)** Given a binary search tree, which of the following **is** or **is not** always sufficient to reconstruct it? For each one, write **YES** if it is enough to reconstruct the tree, or **NO** if it is not. If you answer **NO**, please provide a counter-example to justify.

i) Pre-order traversal. ii) In-order traversal. iii) Post-order traversal.

5. **(6%)** Given a weighted graph $G = (V, E)$, we arbitrarily partition the nodes into two disjoint sets, $V1$ and $V2$. Let $E1$ be all the edges with both endpoints in $V1$; let $E2$ be all the edges with both endpoints in $V2$; let $E3$ be all the edges (u, v) such that $u \in V1$ and $v \in V2$. For example, Figure 1 is the input weighted graph G, and Figures 2 and 3 present the two possible partitions.

If we construct a Minimum Spanning Tree $T1$ on $(V1, E1)$ and a Minimum Spanning Tree $T2$ on $(V2, E2)$, then we connect $T1$ and $T2$ with the lowest-weighted edge in $E3$, will the resulting subgraph be a Minimum Spanning Tree of G?

Please provide a proof if your answer is YES, or give a counterexample if your answer is NO.

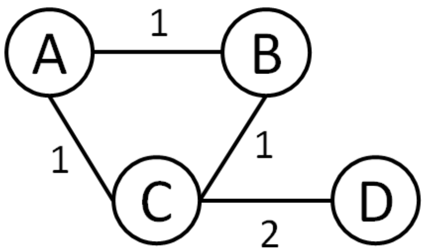

Figure 1. The graph *G*.

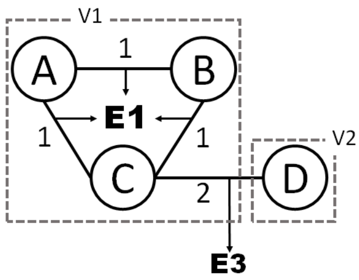

Figure 2. Partition #1.

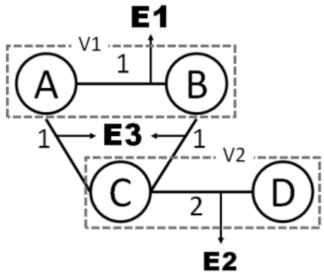

Figure 3. Partition #2.

6. **(12%)** Consider an undirected graph *G = (V, E)* with nonnegative edge weights *w(i, j)* ≥ *0* on edge *(i, j)* ∈ *E*. Let *v* be a node in *G*. Assume you have computed the shortest paths starting from v and the Minimum Spanning Tree of *G*. Given a positive integer *c*, suppose we change the weights on every edge by adding *c* to each of them, i.e., the new weight is *w'(i, j) = w(i, j) + c* for every edge *(i, j)* ∈ *E.*

(a) **(6%)** If *T* is an MST of *G*, will *T* still be an MST of *G* after the change of edge weights? Either give an example if *T* is NOT an MST after the change or prove that *T* is still an MST of *G*.
(b) **(6%)** Would the shortest paths change due to the change in weights? Either given an example if it would change or prove that it will not change.

7. **(7%)** Use stack to implement a queue's *enqueue* (insertion of an element at one end of the queue) and *dequeue* (deletion of an element at the other end of the queue). You need to consider the cases of empty queues and stacks.

Enqueue(e)
{

Your code (*e* is the element to be inserted)

}

Dequeue()
{

Your code

}