Name _____ CS 102 – Final

There are 8 questions. Submit your answers on blackboard as one document (pdf or word). (85 points)

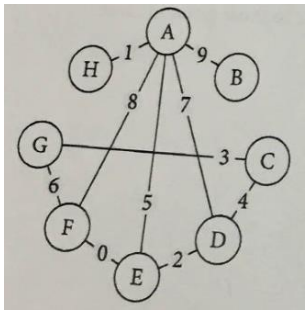
Honor Code Statement for Exam: I, _____, agree to neither give nor receive any help on this exam from other students. I understand that providing answers to questions on this exam to other students is an academic misconduct violation as is taking or receiving answers to questions on this exam from other students. It is important to me to be a person of integrity and that means that ALL ANSWERS on this exam are my answers. Signed _____

1) **(2 pts)** Postfix form the following expression : A + (B * C)
   a) BC*A+ b) ABC*+ c) ABC+* d) BCA*+

2) **(1 pt)** When there is function call then the details of previous function are stored in Stack?
   a) True b) False

3) **(2 pts)** Stack data structure can't be used for
   a) Implementation of recursive function
   b) Allocation resources and scheduling
   c) Reversing string
   d) Evaluation of string in postfix form

4) **(40 pts)** Consider the following undirected weighted graph G:



   a) Represent G using an adjacency matrix (use infinity symbol for unconnected vertices)
   b) Represent G using an adjacency list.
   c) Perform a depth-first search (DFS) traversal of graph, showing: the total order of visit, and the internal stack used by the traversal algorithm. (Start at node A)

d) Perform the breadth-first search (BFS) traversal, showing: the total order of visit, and the internal queue used by the traversal algorithm. (Start at node A)

Notes: in both traversals, whenever a node has to be chosen arbitrarily, choose the node lexicographically smaller (A, is smaller than B, B is smaller than C, etc.)

5) Consider a binary search tree of integers, ordered traditionally (i.e. smaller values to the left, larger values to the right).

a) **(5 pts)** Draw the binary search tree that results from adding the following items to an empty tree, in the order shown below:

25, 12, 8, 32, 46, 17, 15, 30, 27, 10, 11, 9, 5, 52, 55, 50, 39

b) **(5 pts)** Draw the binary search tree that results from removing the value 32 from the tree you drew above.

c) **(10 pts)** Considering the tree resulting from part (b) and list the tree nodes as visited in the *preorder traversal* and *postorder traversal*.

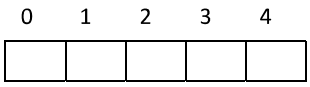
6) **(5 pts)** Consider an hashtable HT with a capacity of 5 items, the hash function hash

Private int hash1(int key) { return key % HT.size; }

Insert the following keys in the table

HT.insert (53); HT.insert (18); HT.insert (4); HT.insert (12); HT.insert (24);

Suppose **HT** uses linear probing, show HT after performing all the insertions above (left to right)



For the following questions 7 and 8 below, use the following declarations for a class of dynamically allocated binary search tree of positive integers.

```
class TreeNode{
  int datum;
  TreeNode left, right;
  public TreeNode () { datum = 0; left = null; right = null; }

  public int getDatum () { return datum; }
  public void setDatum (int val) {datum = val; }

  public TreeNode getLeft () { return left; }
  public void setLeft (TreeNode link) { left = link; }

  public TreeNode getRight () { return right; }
  public void setRight (TreeNode link) { right = link; }
}
```

```
class Tree {
  TreeNode root;
}
```

7) **(5 pts)** Consider the following Java methods belonging to the Tree class. They compile correctly; however, they do not function as described in the comments. What is wrong? How would you fix it?

```
// calls private recursive method to find an int in the tree
public boolean search (int traget){
  return search(target, root);
}
// recursively searches for 'target' starting at 'current'
private boolean search (int target, TreeNode current){
  if (current.getDatum() == target) return true;
  if (current.getDatum() < target)
    return search(target, current.getRight());
  else return search(target, current.getLeft());
}
```

8) **(10 pts)** Write a Java method in the Tree class with the following header:
Private void printRange(TreeNode current, int low, int high)

The method should print all of the values in the binary search tree rooted at "current", whose values are between "low" and "high" (inclusive of both limits), in order.

Note: your method should use the fact that the argument node is the root of a binary search tree to execute "efficiently".