Rutgers Honor Pledge

Rutgers honor pledge: *On my honor, I have neither received nor given any unauthorized assistance during this examination.*

You are on your honor to complete this quiz without assistance. You MAY NOT ask any human for assistance or use any online tutoring service.

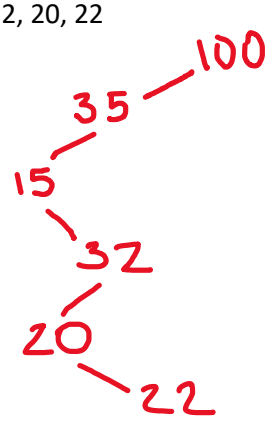
By continuing, you are AGREEING TO THE ABOVE HONOR STATEMENT and Rutgers Academic Integrity policies.

Problem 1 – Miscellaneous (30 points)

a) (10 points) Suppose a BST stores integers in the range from 1 to 100. Suppose that we search for the value 22 and print the values encountered on the search path through the BST. Are the following sequences possible search paths through the BST? If yes, show the path (with branches), if not, explain why.

- (2.5 points) 100, 40, 20, 30, 23, 35, 22
  Cannot have 35 after 23 because 23 is the left subtree of 30 (all values must be smaller than 30)

- (5 points) 100, 35, 15, 32, 20, 22



- (2.5 points) 50, 60, 90, 40, 75
  Cannot have 40 after 90 because is on the right subtree of 60 which is on the right subtree of 50 (all values must be greater than 50).

b) (11 points) For this question you will write two different algorithms that perform the same task. The task is as follows: given an unsorted array of *n* integers find and print items that are duplicates. For example, if the array is [5, 1, 4, 5, 7], the algorithm prints the number 5. Describe your algorithm using pseudocode or a high-level description, do not use Java. Use standard techniques learned in class.

- (5 points) Describe the first algorithm to perform the task mentioned above. This is algorithm expected to run in $O(n^2)$.
  Compare each item against all other items to find duplicates. $O(n^2)$

- (6 points) Describe the second algorithm to perform the task mentioned above. This algorithm is expected to run in O(n log n).
  For each array item
  Insert item into a red-black tree, duplicates will be found when attempting to insert.

  There are n items and each red-black tree insert takes O(log n) in the worst-case, total time is O (n log n)

  OR

  Sort the array using mergesort, heapsort, or quicksort. O(n log n)
  Once the array is sorted, linearly search the array for duplicates. O (n)
  Total O (n log n)

c) (9 points) Fill in the table below with the worst-case scenario running time for each operation. Use big-O notation. The operations are:
   - insert: inserts a new item into the data structure.
   - successor: given an item, return the successor of that item.
   - getMinimum: return the value of the minimum item in the data structure.

Data Structure	**insert**	**successor**	**getMinimum**
Sorted array	O(n)	O(log n)	O(1)
Balanced tree (red-black tree)	O(log n)	O(log n)	O(log n)
Hash table	O(n)	O(n)	O(n)

// 1 point each

Problem 2 – Priority queue (16 points)

Assume a binary heap is stored in an array called `treeNodes`, which has a capacity of 100 array elements, with array index 0 not being used. If the binary heap currently contains 86 keys, answer the following questions.

a) **(4 points)** Is `treeNodes[44]` a leaf node? explain.
Yes; if it is a non-leaf node, children nodes would be at index 44*2 and 44*2+1, that is, treeNodes[88] and treeNodes[89]. The binary heap contains only 86 keys, the last array element would be treeNodes[86].

b) **(4 points)** Does `treeNodes[43]` have only one child node? explain.
Yes; left child is treeNodes[86], which is the last element.

c)

(2 points) (i) Is the subtree rooted at treeNodes[8] a full binary tree? explain.
Yes; follow along the right subtree, the keys would be treeNodes[17], treeNodes[35], and treeNodes[71]; treeNodes [71] is a leaf node without children nodes.
(2 points) (ii) How many levels are there in the subtree rooted at treeNodes[8] including (treeNodes[8])? 4

d)

(2 points) (i) Including the root node (treeNodes[1]) how many levels are there in the binary heap? 7
(2 points) (ii) How many leaf nodes are there? 43

Problem 3 – Hash table (14 points)

a) **(2 points)** Suppose that the keys A through G, with the hash values given below, are inserted in some order into an initially empty table of size 7 using a linear-probing table with no resizing. What is the worst-case running time for a random key search, either hit or miss? O(n) or 7 compares

key	A	B	C	D	E	F	G
hash value	2	2	2	2	2	2	2

b) **(12 points)** Suppose that the keys A through G, with the hash values given below, are inserted in some order into an initially empty table of size 7 using a linear-probing table with no resizing.

key	A	B	C	D	E	F	G
hash value	2	0	0	4	4	4	2

(6 points) (i) What is the number of array accesses required to build the hash table? Justify your answer.

- 12 array accesses **(2 points)**
- (4 points)
  - (1 point) for hash values 0, 1 + 2 = 3 accesses (1 access for inserting B or C, and 2 accesses for inserting the other key)
  - (1 point) for hash values 2, 1 + 2 = 3 accesses (1 access for inserting A or G, and 2 accesses for inserting the other key)
  - (2 points) for hash values 4, 1 + 2 + 3 = 6 accesses (1 access for inserting the first key of D, E or F, 2 accesses for the next key, and 3 array accesses for the next)

(6 points) (ii) When inserting the keys, is the insertion sequence E F G A C B D possible? Justify your answer.

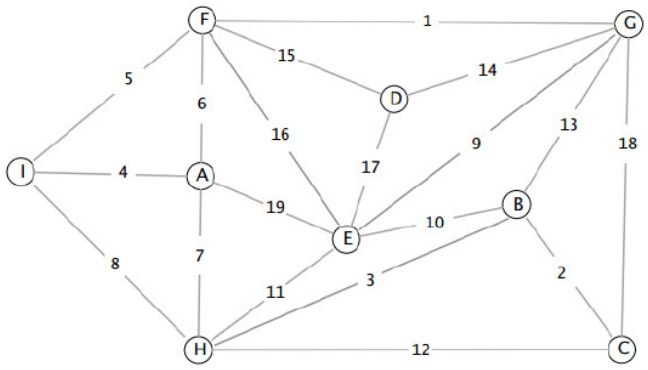
- Yes **(2 points)**
- **(4 points)** the explanation can be one or both of the following bullets:
  - Regardless of the insertion order, there will be 3 clusters,
    - index 0, 1
    - index 2, 3 and
    - index 4, 5, 6

    The hash collision does not overlap from one hash value to another
  - The resulting hash table would be

index	0	1	2	3	4	5	6
key	C	B	G	A	E	F	D

Problem 4 – Undirected Graphs (20 points)

a) Consider the following weighted undirected graph where distances and edges between specific vertices are shown.



(i) (6 points) Starting with vertex I, find the DFS sequence of nodes using the heuristic "*the smaller weight*" to decide which node to select next. For example, if you use the heuristic *"lowest alphabet letter"*, you will go from I to A first.

I ➔ A ➔ F ➔ G ➔ E ➔ B ➔ C ➔ H ➔ D. // no partial credit

(ii) (2 points) What is the total length of the path from I to C? 32

(iii) (4 points) Do you think the answer in part (b) is the shortest path from I to C in this graph? If the answer is *"yes"*, then write few sentences to justify why you think so. If the answer is *"no"*, state why you think the algorithm cannot be used to find the shortest path.

No. It is possible that, although, we choose A as the next node to visit from I based on the heuristic "the smaller weight", there is a shorter distance path I ➔ H ➔ C

b) (8 points) Assume that an undirected graph is stored in an adjacency matrix. Complete the method countEdges in the Graph class that returns the number of edges in the Graph.

```
public class Graph {
       boolean[][] adjmat;
       public int countEdges() { // computes and returns number of edges
}
 public int countEdges () {
    int count = 0;
     for (int i = 0; i < adjmat.length; i++ ) { // 1 points
        for (int j = 0; j < i; j++ ) {          // 1 points
            if ( adjmat[i][j] ) {               // 3 points
               count++;                         // 2 points
            }
        }
     }
     return count;                              // 1 point
}
```

Problem 5 – Directed Graphs (30 points)

a) **(15 points)** A strongly connected directed graph is one in which every vertex can reach all other vertices via paths with one or more edges. List the steps of an algorithms/pseudocode to tell if a directed graph is strongly connected or not. There is no need to explicitly write graph algorithms that we have seen in class. For example, if your algorithm requires DFS or BFS you can simply state call DFS/BFS.

```
[3 pts] for each vertex in the graph
[4 pts]     initialize marked array to all false // has to happen inside the
loop
[3 pts]     call dfs (or bfs) starting at that vertex
[4 pts]     scan visited array, and if a vertex is still unvisited, return
false
[1 pts] return true // after loop is done
```

b) **(15 points)** You are asked to construct the complement of a directed graph with *n* vertices and *e* edges that is stored in an adjacency list. The complement of a graph is a graph that has the same number of vertices as the original graph and has edge $i \rightarrow v$ if there was no edge $i \rightarrow v$ in the original graph.

Write the method `complement` as a method in the DiGraph API used in class. The method returns a graph that is the complement of the original graph in the DiGraph class.

public class Digraph

	Digraph(int V)	*create an empty digraph with V vertices*
	Digraph(In in)	*create a digraph from input stream*
void	addEdge(int v, int w)	*add a directed edge v→w*
Iterable<Integer>	adj(int v)	*vertices adjacent from v*
int	V()	*number of vertices*
int	E()	*number of edges*
Digraph	complement()	*complement of this digraph*

```
public Digraph complement () {
}
```

```
//There could be other solutions

public Digraph complement () {
   int numVertices = V();
   boolean[][] edges = new boolean[numVertices][numVertices];

   // create boolean array with original graph's edges

   // 5 points: student uses a data structure to find original
   // edges to later on create their complement

   for ( int v = 0; v < numVertices; v++ ) {
     for ( int w : adj(v) ) {
       edges[v][w] = true;
     }
   }

   // create the complement graph
   Digraph C = new Digraph(numVertices); // 2 points

   // add an edge for every edge not present in the original graph
   for ( int v = 0; v < numVertices; v++ ) {
     for ( int w = 0; w < numVertices; w++ ) {
       if ( edges[v][w] == false ) {
         C.addEdge(v,w); // adds complement edge 7 points
       }
      }
    }
    return C;   // 2 points
}
```

Problem 6 – Sort (20 points)

a) **(2.5 points)** When is insertion sort a good choice for sorting an array?

   a. Each array item requires a large amount of memory.
   b. Each array item requires a small amount of memory.
   c. The array has only a few items out of place.
   d. The processor speed is fast.

b) **(2.5 points)** Suppose that we are sorting an array of eight integers using a quadratic sorting algorithm. After four iterations of the algorithm's outer loop, the array elements are ordered as shown here: 2 4 5 7 8 1 3 6

   Which statement is correct? Note that our selection sort picks largest items first.

   a. The algorithm might be either selection sort or insertion sort.
   b. The algorithm might be selection sort, but it is not insertion sort.
   c. The algorithm is not selection sort, but it might be insertion sort.
   d. The algorithm is neither selection sort nor insertion sort.

c) **(2.5 points)** Mergesort makes two recursive calls. Which statement is true after these recursive calls finish, but before merge?

   a. The array items form a heap.
   b. Items in each half of the array are sorted amongst themselves.
   c. Items in the first half of the array are less than or equal to the elements in the second half of the array.
   d. None of the above.

d) **(2.5 points)** Describe a case using an example where quicksort will result in quadratic behavior.

   Whenever the partition function repetitively divides the array in two halves of sizes 0 and n-1 respectively. That happens when the array items are in increasing or decreasing order.

1	10	12	14	15	28	37	60	79

e) **(5 points)** The following array has just been partitioned by the first call to quicksort partition function:

3	0	2	4	5	8	7	6	9

   Which of these elements could be the pivot? Note that there may be more than one possibility.

   4, 5, 9

f) **(5 points)** Show the following array after one call to the quicksort partition function has finished.

5	3	8	9	1	7	0	2	6	4

// 0.5 points for every correct item

0	3	4	2	1	5	7	9	6	8