CS314 Fall 2023 Exam 3 Suggested Solution and Grading Criteria.

Grading acronyms:
AIOBE - Array Index out of Bounds Exception may occur.
BOD - Benefit Of the Doubt. Not certain code works, but, can't prove otherwise.
Gacky or Gack - Code very hard to understand even though it works. (Solution is not elegant. Lack of Zen.)
LE - Logic Error in code.
MCE - Major Conceptual Error. Answer is way off base, question not understood based on answer provided.
NAP - No Answer Provided. No answer given on test.
NN - Not Necessary. Code is unneeded. Generally, no points off.
NPE - Null Pointer Exception may occur.
OBOE - Off By One error. Calculation is off by one.
RTQ - Read The question. Violated restrictions or made incorrect assumption.
EFF - Efficiency. Order is worse than expected or unnecessary computations done.

1. Answer as shown or -2 unless question allows partial credit.
First use of quotes in output is wrong, then error carried forward.
No points off for minor differences in spacing, capitalization, commas, and braces.
Text in parenthesis not required. It is simply grading guidance and / or a brief explanation for answer.

A. 43
B. $12N^2 + 16N + 20$, +/- 2 on each coefficient
C. 2047
D. .0004 seconds (code is O(N))
E. 9 seconds
F. 5 1 5 5
G. 60 seconds
H. 27 seconds (code is $O(N^2)$)
I. 6 1 8 7
J. {0=5, 3=4, 5=2}
K. 9 7 6 6
L. A, C, E (no partial credit)
M. 0.4 seconds (code is O(N))
N. 12 (every path to a node could be a word)
O. 109 bits (10per leaf 1 per internal node)
P. SCF: YES STF: NO (1 point each)
Q. min: 0 or 2 max: 6 (1 point each)
R. 7 6 8 1
S. 17 8 17 3 8 12
T. 44 seconds
U. No
V. A
W. Yes
X. H F G B C D (H was given)
Y. [10, 3, 6]

EXTRA CREDIT +1 Point Each

EXTRA CREDIT 1: Alaska (USS OK)

EXTRA CREDIT 2: Taylor Hall (Just Taylor okay)

2.
```
    public int deepestDepth(int tgt) {
        return deepestDepth(root, 0, tgt);
    }

    private int deepestDepth(BNode<E> n, int currentDepth, E tgt) {
        if (n == null) {
            return -1;
        }
        int depthHere = -1;
        if (tgt.equals(n.data)) {
            depthHere = currentDepth;
        }
        int leftDepth = deepestDepth(n.left, currentDepth + 1, tgt);
        int rightDepth = deepestDepth(n.right, currentDepth + 1, tgt);
        return Math.max(Math.max(depthHere, leftDepth), rightDepth);
    }
}
```

17 points, Criteria:
OKAY TO ASSUME NO NULL VALUES STORED

- create helper, 1 point
- call helper with root, initial depth, and target, 1 point (lose if parameter for max)

For Helper

- base case checked, 2 points
- return -1 in base case, 2 points
- recursive case, check if target here with equals, 2 points
- if current node stores target considers current depth, 2 points
- correctly increase depth when going to children, 2 points
- recursive calls to left and right correct, 3 points
- correctly check for max between three possible options 1 point
- return correct result, 1 point

Other deductions:

- array of length 1 to track answer, - 4 (disallowed in question)
- Parameter for max logic error, -4
- Alter tree, -5
- NPE -5
- early return -6
- doesn't track depth -4
- no tracking of depth -6
- no comparison of current to left and right, -5
- infinite loop, -6
- parameter for best but not returning that, -1 + -3
- 2N, -2 (efficiency)

3. Comments:

```
    public TreeSet<String> getSourceVertices() {
        for (String vertexName : vertices.keySet()) {
            vertices.get(vertexName).scratch = 0;
        }

        // Determine indegree of each Vertex.
        for (String vertexName : vertices.keySet()) {
            Vertex source = vertices.get(vertexName);
            for (Edge e : source.adjacent) {
                e.dest.scratch++;
            }
        }

        TreeSet<String> sources = new TreeSet<>();
        for (String vertexName : vertices.keySet()) {
            Vertex v = vertices.get(vertexName);
            if (v.scratch == 0) {
                sources.add(vertexName);
            }
        }

        return sources;
    }
```

16 points, Criteria:

- iterator through keyset correctly, 2 points
- access Vertex value via get correctly, 1 points
- 0 out scratch variables before determining indegree (must be separate loop), 2 points (lose if call clearAll)
- loop though all vertices to determine in degree, 1 point
- loop through edges of current vertex, 1 point
- increment scratch or set scratch to non flag (typically 0) value for destination of current edge, 2 points
- create result, 1 point
- last loop to find sources, 1 point
- correctly check if indegree is 0 (or some correct flag value), 2 points
- Add source vertices to result, 1 point
- return result, 1 point

Other:

- Disallowed methods, varies (remove on set -3)
- Add or remove any vertices or edges, -5
- recursion -5
- O(V^2), -5
- read bit at a time, manually convert to base 10, -2 (efficiency)
- other data structure created besides resulting TreeSet, -4

4. Comments:

```
public static ArrayList<Integer> decodeDeltaEncoding(BitInputStream
                                                bis) throws IOException {
    final int DELTA_SIZE = bis.readBits(5);
    // Read the first value.
    int value = bis.readBits(32);
    ArrayList<Integer> result = new ArrayList<>();
    result.add(value);
    // Fence post. Ugh.
    int signBit = bis.readBits(1);
    while(signBit != -1) {
        // Get the next delta
        int delta = bis.readBits(DELTA_SIZE);
        delta = signBit == 0 ? delta : -delta;
        value += delta;
        result.add(value);
        // Get the next sign bit if it exists
        signBit = bis.readBits(1);
    }
    return result;
}
```

17 points, Criteria:

- read and store num bits in each delta, 2 points (lose if assume +1 to read value)
- read first value, 1 point
- read first sign bit (fencepost), 1 point
- while loop that checks value of last read != -1, 4 points
- read delta, 1 point
- correctly determine and apply sign bit, 3 points
- determine next value in sequence based on delta and previous value, 2 points
- add value to result, 1 point (lose if forget to add first value)
- read next sign bit, 1 point (could use if for next delta as well)
- create and return result, 1 point

Other:

- ArrayList get, -1
- ArrayList size, -1
- Treat BitInputStream as array or list, -4
- manually convert from base 2 to base 10, -3
- just readBits not bis.readBits, -3