**NAME:**

**COMP231 – Data Structures and Problem Solving**

**Spring 2018**

**Midterm Exam #1**

**March 8, 2018**

1. Consider the following method:

**public** **boolean** majorityEven(**int**[] list) {

**int** count = 0;

**for**(**int** i = 0; i< list.length; ++i) {

**if**(list[i]%2 == 0) {

count++;

}

**if**(count > list.length/2) {

**return** **true**;

}

**if**(count + list.length-i < list.length/2) {

**return** **false**;

}

}

**return** **false**;

}

[2] a. What is the problem size, n, for this algorithm?

[4] b. Describe the input that will evoke the best-case running time for this algorithm.

[4] c. Describe the input that will evoke the worst-case running time for this

algorithm.

[5] d. Counting array accesses as the basic operation, give an equation T(n) for the worst-case operation count for the method.

2. Consider the following method:

**public** **int** functionX(**int** x) {

**int** i = 0;

**while**(x%2 != 0) {

i += 1;

x = x /2;

}

**return** i;

}

[1] a. What is the problem size, n, for this algorithm?

[4] b. Give the best Ω bound for the running time of the functionX method.

[4] c. Give the best Ο bound for the running time of the functionX method.

3. Consider the following recursive method:

**public** **int** functionY(**int**[] x, **int** index) {

**if**(index == x.length -1) {

**return** x[index];

}**else** {

**return** x[index]\* functionY(x, index+1);

}

}

Counting array access as the basic operation, give a recurrence relation for the above method.

[2] T(0) =

[4] T(n) =

4. Consider the following recurrence relation giving the basic operation count for a recursive method:

T(n) = T(n/2) + n2 - 1

T(0) = 7

[3] a. Show the first step in the use of the expansion technique for solving this

recurrence.

T(n) =

[4] b. Continue using the technique of expansion to give a closed form expression for

T(n).

[ 7] 7. Consider the following basic operation count for an algorithm:

T(n) = 5n2+ 14n lg n + 25

Use the definition of Ο to show that T(n) is in Ο(n2)

[8] 8. Assuming that A and B are two algorithms for solving the same problem, circle T or

F to indicate whether each of the following statements is true or false.

1. T / F If A is in Ω(n) and B is in O(lg n), then A will always run slower than B.
2. T / F If A is in Ω(n) and B is in Ω(nlg n) then A will always run faster than B
3. T / F If A is in Ω (1), then the best case input for A is when the input is a small size.
4. T / F If A is in Ο(n3) and B is in Ο(n2) then there must be an instance of the problem for which B will run faster than A.
5. T / F If A is in O(n) and B is in Ω (n lg n), then the problem is O(n2­)
6. T / F If A is in Ο(n) and B is in Ω(lg n) then there must be an instance of the problem for which B will run faster than A.
7. T / F If A is in Θ(n) and B always runs slower than A, then B is not in Θ (n).
8. T / F If A is in Ω(n) and also in Ο(n lg n) and B is in Θ(n2), then there must be an instance of the problem for which A will run faster than B.
9. T / F If the problem is Θ(n2), then A’s best case is Ω(n2)
10. T / F If A is in Ω(n) and B is in O(lg n), then there must be an instance where A runs slower than B.
11. T / F If A is in O(n) and B is in Ω (n lg n), then there must be an input size k where A is always faster for inputs bigger than k.

9. The lOBISS method finds the length of the biggest increasing subsequence (ordered subset) of a list of numbers. For example:

[4, 2, 1, 5, 10, 3] → 3

The subsequence is [4, 5, 10] or [2, 5, 10] or [1, 5, 10]

[5, 8, 1, 10, 2, 3, 4] → 4

The subsequence is [1, 2, 3, 4]

[5] a. Complete the lOBISS method shown below assuming that the lOBISSHelper method defined in part b works.

public static boolean lOBISS (int[] vals) {

}

[15] b. The lOBISSHelper method below solves a transformed version of the original problem. Complete the implementation of lOBISSHelper so that the lOBISS method from part a will answer the original question.

public static boolean sumNoConsecutiveEvensHelper (int[] vals, int curIndex, int lastNumberIncluded) {

11. For the tree given above, answer the following questions (2pts each):

What type of node is F? \_\_\_\_\_\_\_\_\_\_\_

What type of node is A? \_\_\_\_\_\_\_\_\_\_\_

What is the left child of Y? \_\_\_\_\_\_\_\_\_\_\_\_

How many nodes does the subtree rooted at B have? \_\_\_\_\_\_\_\_\_\_\_

What is the distance from O to A? \_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is the width of the tree? \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Give the postorder traversal of the tree. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**public** **void** visitGregoryOrder(BTNode<K, V> node) {

**if**(node == **null**) {

**return**;

}

System.out.println(node.key);

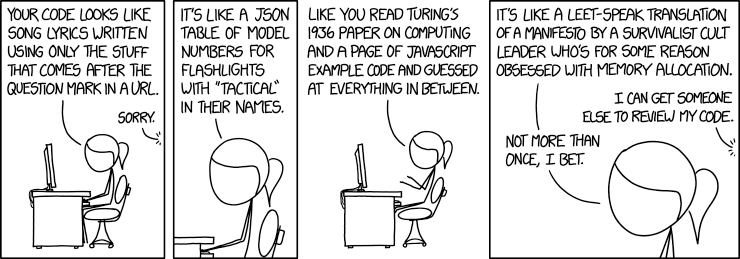
visitGregoryOrder(node.left);

System.out.println(node.key);

visitGregoryOrder(node.right);

}

What would the above code print when given the root as its initial input?



Bonus: Guess your percentage score on this exam (out of 100, before any bonus is applied to your score)

Bonus: Prove

Bonus: Draw all binary trees with 4 nodes.

Bonus: Find a theta value for