# Computer Science II

# Spring 2018

# Final Exam Solutions

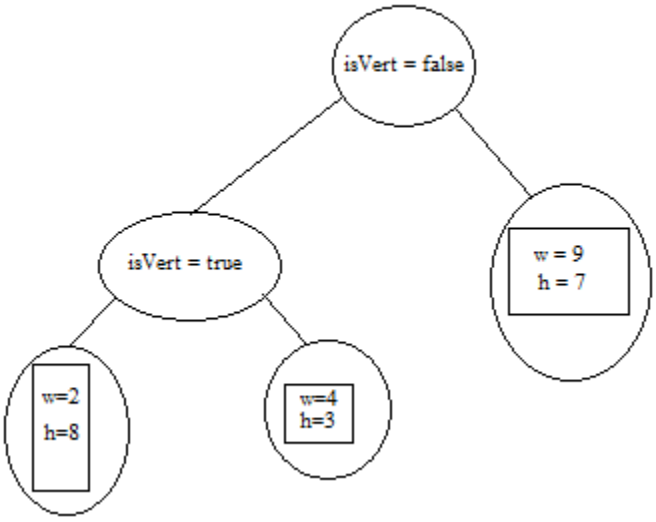
# Instructor: Arup Guha

# Date: 5/1/2018

1) (15 pts) In two guest lectures, Dr. Ewetz described an open problem dealing with rectangle packing. A single packing operation combines two rectangles with dimensions $w_1$ x $h_1$ and $w_2$ x $h_2$. We can choose to combine these two rectangles in two ways:

1. Stack on top of each other to create a rectangle of size $\max(w_1, w_2)$ x $(h_1 + h_2)$
2. Stack next to each other to create a rectangle of size $(w_1 + w_2)$ x $\max(h_1, h_2)$

Furthermore, for a particular arrangement of rectangles, we can store it in a tree structure, where leaf nodes are the original rectangles and internal nodes represent combining two rectangles in one of these two fashions. Consider the following tree:




In this tree, the non-root internal node represents packing a 2 x 8 and 4 x 3 rectangle vertically, which will yield a 4 x 11 rectangle. Then, the root node, represents packing a 4 x 11 and 9 x 7 rectangle horizontally to yield a 13 x 11 rectangle.

For this problem, you will write a method getDim() in the rectangle class that will return an integer array of size 2 storing the width and height, respectively, of the rectangle object (a node in this tree) that the method is called on. The class has the following instance variables:

```
private int w;
private int h;
private boolean isLeaf;
private boolean isVert;
private rectangle left;
private rectangle right;
```

If the object represents a leaf node, then w and h will store the width and height of the rectangle, respectively and isLeaf will be set to true. If the object represents an internal node, then isLeaf will be set to false, isVert will be set to whether or not the two subtree rectangle should be packed vertically or not (true for vertical), and left and right will point to the left and right subtrees, respectively.

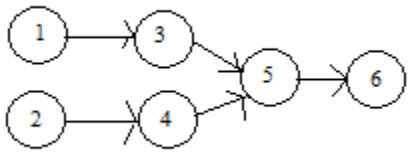
Complete the code for the getDim() method in this class. Note: You may call the method Math.max, which takes in two integers and returns the larger of the two.

```
public int[] getDim() {
    if ( isLeaf ) return new int[]{w,h};     // 1 pt
    int[] leftdim = left.getDim() ;          // 2 pts
    int[] rightdim = right.getDim() ;        // 2 pts
    if ( isVert )                            // 1 pt

        return new int[]{ Math.max(leftdim[0], rightdim[0]), // 2 pts
                          leftdim[1]+rightdim[1]};           // 2 pts
    else

        return new int[]{ leftdim[0]+rightdim[0],            // 2 pts
                          Math.max(leftdim[1], rightdim[1])};// 2 pts
     }
}     // 1 pt added for getting everything
```

2) (10 pts) Below is a 6 line code segment in Python that prompts the user to the # of cups of sugar they have and the number of lemons. These figures will be used to calculate the maximum number of pitchers of lemonade that can be made from these ingredients. For these lines of code, find each direct dependency (ie line 3 must go before line 5) necessary for the code to run properly. Draw the corresponding graph using each line as a vertex and each dependency as an edge. Then, count the number of possible topological sorts of this graph and explain the significance of this number.

```
lemons = int(input("How many lemons?\n"))                    // line 1
sugar = int(input("How many cups of sugar?\n"))              // line 2
limitLemons = lemons//3                                      // line 3
limitSugar = sugar*4                                         // line 4
pitchers = min(limitLemons, limitSugar)                      // line 5
print("You can make",pitchers,"pitchers of lemonade.")       // line 6
```

Graph




**Grading: 5 pts - 1 pt for each correct edge, -1 pt for each added incorrect edge, cap at 0.**

Number of Top Sorts: **6** **(Grading: 3 pts correct, 2 pts - off by 2, 1 pt if off by 4, 0 otherwise.)**

Significance of # of top sorts: **Number of different orders the lines of code above could be ordered to produce a correct program.** **(Grading: 2 pts, can give partial)**

3) (15 pts) Determine the fewest number of multiplications to calculate the product ABCDE, for matrices A, B, C, D, E with the following dimensions:

| Matrix | Dimensions |
|---|---|
| A | 2x5 |
| B | 5x1 |
| C | 1x4 |
| D | 4x6 |
| E | 6x3 |

In order to get full credit you must fill out the chart below appropriately, as shown in class. Please include your calculations below the chart.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 | **10** | **18** | **46** | **58** |
| B | X | 0 | **20** | **54** | **57** |
| C | X | X | 0 | **24** | **42** |
| D | X | X | X | 0 | **72** |
| E | X | X | X | X | 0 |

**AB = 2 x 5 x 1 = 10, BC = 5 x 1 x 4 = 20, CD = 1 x 4 x 6 = 24, DE = 4 x 6 x 3 = 72**

**ABC = (AB) x C = 10 + 2 x 1 x 4 = <u>18</u>**
**A x (BC) = 20 + 2 x 5 x 4 = 60**

**BCD = (BC) x D = 20 + 5 x 4 x 6 = 140**
**B x (CD) = 24 + 5 x 1 x 6 = <u>54</u>**

**CDE = (CD) x E = 24 + 1 x 6 x 3 = <u>42</u>**
**C x (DE) = 72 + 1 x 4 x 3 = 84**

**ABCD = A x (BCD) = 54 + 2 x 5 x 6 = 114**
**= (AB) x (CD) = 10 + 24 + 2 x 1 x 6 = <u>46</u>**
**= (ABC) x D = 18 + 2 x 4 x 6 = 66**

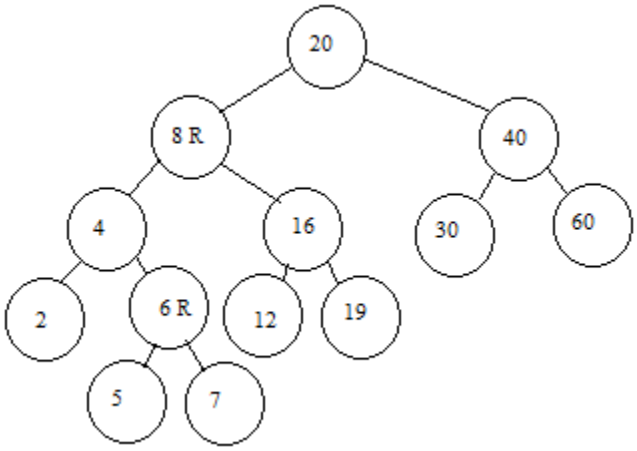
**BCDE = B x (CDE) = 42 + 5 x 1 x 3 = <u>57</u>**
**(BC) x (DE) = 20 + 72 + 5x4x3 = 152**
**(BCD) x E = 54 + 5 x 6 x 3 = 144**

**ABCDE = A x (BCDE) = 57 + 2 x 5 x 3 = 87**
**(AB) x (CDE) = 10 + 42 + 2 x 1 x 3 = <u>58</u>**
**(ABC) x (DE) = 18 + 72 + 2 x 4 x 3 = 114**
**(ABCD) x E = 46 + 2 x 6 x 3 = 82**

**Grading: 1 pt each for AB, BC, CD, DE**
**1 pt each for ABC, BCD, CDE**
**2 pts each for ABCD, BCDE**
**4 pts for ABCDE**

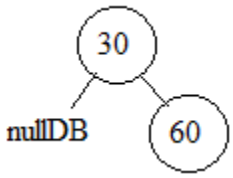
**Can give partial - take off 1 pt per error, try to deal with cascading errors, but not necessary to do so**

4) (10 pts) Draw the result of deleting the value 40 from the red-black tree shown below. In the drawing below, red nodes are indicated with a letter 'R' next to the number stored in the node. In your solution, please put an 'R' in each node that is red. (Note: use the regular binary tree rules for deleting a value which is stored in a node with 2 children.) If you explain your thinking, you may get partial credit for incorrect solutions.

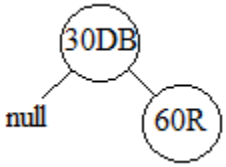



**Note: When deleting 40, we can either delete the physical location of node 30 or 60. Due to the length of the solution, this solution only shows the result of deleting the physical location of the 30. A second correct answer exists (and was properly graded for) where the physical node storing 60 is deleted.**

**After we denote the physical node storing 30 to be deleted, we copy 30 into the node that originally stored 40. Here is the state of affairs from that point, just for the subtree rooted at 30:**

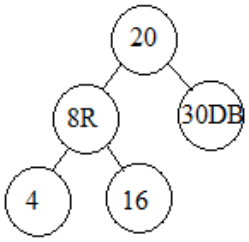



**Initially, the null node is a double black.**

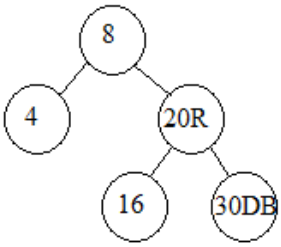



**We push the double black up one level.**

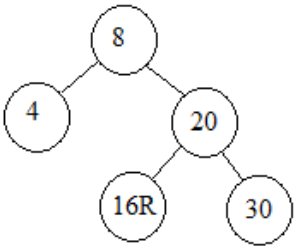
**Then, we deal with a double black that has a red sibling:**




**This is the original picture.**

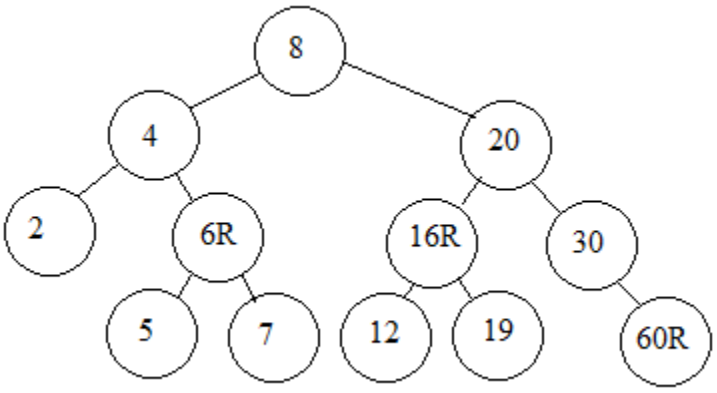



**Here is when we right rotate making 8 the root and 20 red.**




**Push the double black up one level, fixing the issue.**

**Here is the final tree after those changes are made:**




**Note: if we had deleted the physical node storing 60 instead of the one storing 30, the only difference in the final answer is that 60 would be a black child of 20 and 30 would be a red left child of 60.**

**Grading: Give full credit if final tree is correct. If incorrect, give partial as follows:**

**1 pt - deleting physical node of 30 or 60.**
**2 pts - db null node**
**2 pts - pushing up db to 30 or 60.**
**3 pts - restructuring with 8 at root, 4 to left and 20 to right**
**2 pts - push up double black**

5) (14 pts) The country of PaperTrailLandia runs its national presidential election via a very simple paper voting system. Each citizen submits a single piece of paper with the name of a single person, for whom they wish to vote. For the purposes of this problem, we assume that each name is a string of uppercase letters and that each distinct person has a distinct name. Complete the method below which takes all pieces of paper as a String array, and returns a HashMap<String,Integer> which maps each person receiving a vote to the number of votes they received.

```
public static HashMap<String,Integer> getMap(String[] names) {

    HashMap<String,Integer> map = new HashMap<String,Integer>();

    for (int i=0; i< names.length; i++) {          // 1 pt

        if ( map.containsKey(names[i]) )           // 3 pts

            map.put(names[i], map.get(names[i]) + 1); // 6 pts

        else

            map.put(names[i], 1);                  // 4 pts

    }

    return map;
}
```

6) (11 pts) In class, a dynamic programming algorithm was covered that efficiently determines the ***fewest*** number of coins necessary to make change for a particular number of cents given the valid denominations of coins (and an infinite supply of each). In this algorithm, the array entry for dp[value] simply stores the fewest number of coins necessary to make change for value number of cents. For this problem, fill in this array for indexes 1 to 22, given that the valid denominations of coins are 1 cent, 3 cents, 8 cents and 14 cents.

| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mincoins | 0 | **1** | **2** | **1** | **2** | **3** | **2** | **3** | **1** | **2** | **3** | **2** |

| index | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| mincoins | **3** | **4** | **1** | **2** | **2** | **2** | **3** | **3** | **3** | **4** | **2** |

**Grading: 1/2 for each slot, round down to next integer**

Questions 7 and 8 of the exam are going to look at analyzing the following game:

You are given n k-sided fair dice, each labeled 1, 2, 3, …, k. You roll all of them. Then, separate out the ones that show k. Take the rest and roll them all again. Then, separate out the ones of these that show k, and roll the rest again. Repeat this process until you've separated all the dice out (ie, they all show k). The question we want to analyze is the number of times we expect to roll before completing the game.

7) (14 pts) One way to analyze this is to write a simulation, run it many times and take the average. The latter portion of this is fairly trivial, so for this question, you will simply write a single Java method that takes in n, the number of dice, and k, the number of sides on each dice, simulates this process, and returns the number of turns it took to complete a single simulation of the game. Fill in the method signature given below. Assume that you have access to a static class variable r, that is of type Random, which you can use to generate random integers.

```
public static Random r;

public static int numTurnsSim(int n, int k) {

     int turns = 0;                          // 1 pt
     while (n > 0) {                         // 1 pt

          int numK = 0;                      // 1 pt

          for (int i=0; i<n; i++) {          // 2 pts
               int die = r.nextInt(k);       // 3 pts
               if (die == k-1) numK++;       // 3 pts
          }

          turns++;                           // 1 pt
          n -= numK;                         // 1 pt
     }

     return turns;                           // 1 pt
}
```

**Grading note: In general, the breakdown is 9 points for running one set of throwing the dice, 5 pts for repeating these sets, counting the number of repeats and returning it.**

8) (10 pts) Let the dice have k sides each and let T(n) equal the expected number of turns to complete the simulation described previous. A recurrence relation that T(n) satisfies is as follows:

$$T(0) = 0$$

$$T(n) = 1 + \sum_{i=0}^{n} [\binom{n}{i} \left(\frac{1}{k}\right)^{i} \left(\frac{k-1}{k}\right)^{n-i} T(n-i)]$$

In words, explain why this formula is correct. In your explanation, please explain what 1 represents, what the summation index i, represents and what T(n-i) represents. (Of course, explain what each part represents and why their interaction is the way that it is.) Note: One issue with this formula is that a T(n) term appears on the right hand side. But, if one were to solve this recurrence, we can easily take care of this problem by subtracting that term to the left hand side and factoring out T(n). Then, we would have a factor by which we could divide both sides of the resulting equation.

**The 1 represents a turn throwing the the n dice. Of these dice, any where from 0 to n of them could turn up to land showing k. The variable i in the summation index represents the number of dice that show k out of the n dice that are rolled. For each of these possibilities, we must calculate the probability of that happening and multiply that by the expected number of turns of the simulation which will now have n - i dice left. The probability that i dice show up with k is based on the binomial distribution. In general, if we repeat a trial n times where the probability of success is p, the probability of getting exactly i success is $\binom{n}{i}(p)^{i}(1-p)^{n-i}$. For this particular problem, the probability of success, ie. rolling a k, is $\frac{1}{k}$. This explains the first three terms in the sum; these are just the product that represents the probability that i of the dice will show k. We must multiply this probability by the expected number of future turns, but this is just T(n - i), since T(x) represents the expected number of turns in the game starting with x die for any non-negative integer x. If i of the dice show k, then that means we continue to play the game with n - i dice, and using the definition of expectation, it follows that the appropriate term to multiply the aforementioned probability by is just T(n - i). This explains the recurrence relation above in full.**

**Grading: 2 pts for explaining the +1.**
**2 pts for representing what i means.**
**2 pts for explaining why we are summing over those possible i.**
**2 pts for mentioning that the first three terms are the desired probability**
**2 pts for explaining that if i dice show k, n-i dice remain and that T(n-i) is the number of turns we expect the game to go, by definition of T.**

9) (1 pt) Who wrote The Autobiography of Mark Twain? **Mark Twain** **(1 pt give to all)**