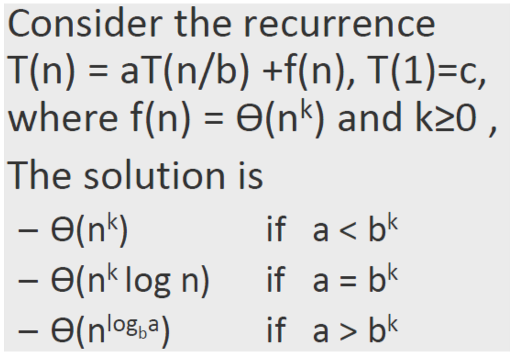
MA/CSSE 473 Exam #2 Fall, 2014 (Day 34) Name____________________ Section: 01(9:55) __ 02(10:50)__

Resources allowed: Calculator, one 8.5 x 11 sheet of paper (one-sided, hand-written).
Resources not allowed: Anything that can communicate or has headphones/earphones.

It is possible to get so caught up in getting all of the points for one problem and spend so much time on it that you do not get to the other problems. Don't do that! I will be generous with partial credit if you have the main ideas. You should first do the problems you are confident about, and then do the rest. Time is likely to be a factor on this exam.

For Instructor use:

Problem	Possible	Earned
1a	7	
1b	4	
1c	5	
2a	8	
2b	8	
3a	6	
3b	6	
3c	4	
4a	8	
4b	8	
5	16	
Total	80	



Consider the recurrence
$T(n) = aT(n/b) + f(n)$, $T(1)=c$,
where $f(n) = \Theta(n^k)$ and $k \geq 0$,
The solution is
- $\Theta(n^k)$ if $a < b^k$
- $\Theta(n^k \log n)$ if $a = b^k$
- $\Theta(n^{\log_b a})$ if $a > b^k$

1.(16) This problem refers to the Optimal Binary Search Tree dynamic programming algorithm from class, where we examined a table similar to the one below.

Input parameters. I spaced the a and b values to try to make it clear which b values are between which a values. Recall that the a's are the frequencies of the successful search (interior) nodes, and b's are the frequencies of the unsuccessful (exterior) nodes.

```
n = 6
letters =     ['A',  'E', 'I', 'O', 'U', 'Y']
a =  [None,     10,   2,   5,   6,   4,   3]
b = [         1,   4,   7,    8,   0,   3,   2]
```

Values in the output table: Recall that when the subscripts are ij, it is referring to a subtree that contains internal nodes i+1, …, j and external nodes i, …, j. In each square, the R value is number of the root of the optimal tree, the C value is the weighted path length of that tree, and the W value is the sum of the a and b values for all of the nodes in that subtree tree.

(a) Show the values of R_{35} ____________ W_{35}__________ C_{35}____________ (points: 2, 2, 3)

(b) (4) In the space below, show how you use the dynamic programming algorithm to get W_{35} and C_{35} with only a few additions.

(c) (5) Also in the space below the grid, draw a diagram of the optimal tree, labeling each node by its letter value.

R00: 0 W00: 1 C00: 0	R01: 1 W01: 15 C01: 15	R02: 1 W02: 24 C02: 37	R03: 2 W03: 37 C03: 72	R04: 3 W04: 43 C04: 94	R05: 3 W05: 50 C05: 115	R06: 3 W06: 55 C06: 137
	R11: 1 W11: 4 C11: 0	R12: 2 W12: 13 C12: 13	R13: 3 W13: 26 C13: 39	R14: 3 W14: 32 C14: 59	R15: 3 W15: 39 C15: 80	R16: 3 W16: 44 C16: 102
		R22: 2 W22: 7 C22: 0	R23: 3 W23: 20 C23: 20	R24: 3 W24: 26 C24: 40	R25: 4 W25: 33 C25: 60	R26: 4 W26: 38 C26: 77
			R33: 3 W33: 8 C33: 0	R34: 4 W34: 14 C34: 14	R35: W35: C35:	R36: 4 W36: 26 C36: 45
				R44: 4 W44: 0 C44: 0	R45: 5 W45: 7 C45: 7	R46: 6 W46: 12 C46: 19
					R55: 5 W55: 3 C55: 0	R56: 6 W56: 8 C56: 8
						R66: 6 W66: 2 C66: 0

2. (16) [This comes from Levitin 8.4.6]

(a) (8) Explain how Warshall’s algorithm can be used to determine whether a given digraph is a dag (directed acyclic graph).

Is this an efficient algorithm for this problem? (Circle one : Yes No) Briefly explain why or why not.

(b) (8) Is it a good idea to apply Warshall’s algorithm to find the transitive closure of an undirected graph?
(Circle one : Yes No)
Briefly explain why or why not.

3. (16) [Levitin 7.2.5, 7.2.6]

(a) (6) Is[CSSE1] it possible that there are a pattern and text such that Horspool’s algorithm does more character comparisons than the brute-
force algorithm would make when searching for the same pattern in the same text? (circle one: Yes No)
If so, give an example of such a pattern and text. If not, prove that it is not possible.

(b) (6) If Horspool’s algorithm finds a matching substring, how far should it shift the pattern to begin searching for the next possible match in the text?

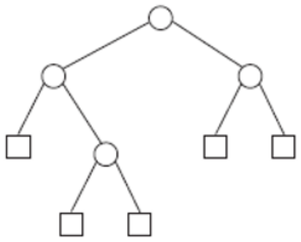
(c) (4) In the KMP algorithm, if the first character of the pattern is compared to a character *c* from the text, and if *c* does not occur anywhere in the pattern, what is the maximum shift that we can do?

4 (16 points) The seven elements of a binary (min) heap are stored in positions 1-7 of an array. If the original arrangement of the array elements is not a heap, we examined two algorithms for making it into a heap. (1) top down, repeated insert of elements 2 through N (with percolate up) to build up the heap, and (2) bottom-up heap construction, using the buildHeap (also known as *fixheap*) bottom-up (with percolate down) algorithm.
If the original array is as shown below, show the results of running each of the two algorithms. Be very careful because it will be difficult to give partial credit. If it helps you get the answers, feel free to draw the binary trees represented by the arrays.

	0	1	2	3	4	5	6	7
Original array		3	7	2	5	6	1	4
Heap after repeated insert on original array								
Heap after buildheap on original array								

5. (16 points) {Levitin 5.3.9] Recall that an ***extended binary tree*** (EBT) T consists of either a single external node, or a root (internal node) and two EBTs T_L and T_R.

The ***internal path length*** (IPL) of an EBT T is defined as the sum of the depths of the internal nodes in T. Similarly, the ***external path length*** (EPL) of T is defined as the sum of the depths of T's external nodes. For example, in this EBT,



IPL(T) = 4 (0 + 1 + 1 + 2)   EPL(T) = 12. (2 + 3 + 3 + 2 + 2)

Prove by induction that for every EBT T containing N internal nodes, EPL(T) = IPL(T) + 2N.

[Suggestion: Use the recursive definition of EBT, and strong induction.]