CS2210b: Data Structures and Algorithms
Winter 2014: Midterm Examination Solutions
120 minutes

Instructions:

- Write your name and student number in the space provided.
- Please check that your exam is complete. It should have 14 pages in total
- The examination has a total of 100 marks.
- This is an open textbook/lecture notes exam. You can consult your notes/textbook but you cannot talk/communicate with other students. No assignment/sample midterm solutions are allowed.
- When you are done, raise your hand and we will pick up your exam.

Name:_____

Student Number:_____

Part 1	
13	
14	
15	
16	
17	
18	
19	
TOTAL	

PART1: Multiple Choice

Instructions: *Enter your answers on the Scantron sheet. We will not mark answers that have been entered on this sheet. Each multiple choice question is worth 3 marks.*

SOLUTIONS: DECBAADBBDDA

1. What is the best asymptotic ("big-O") characterization of the following function:

$$f(n) = 2^5 \ + 5\ n^3\ \log(n) + 2^6\ n^2 + 100\ n^4$$

   (a) $O(n^2)$
   (b) $O(n^3)$
   (c) $O(n^3\ \log n)$
   (d) $O(n^4)$
   (e) $O(n^6)$

2. Let $f(n) = 50n^4 + 10n^2 + 5n$ and $g(n) = 25n^3 + 2n$. Which of the following is **not true**:

   (a) $f(n) + g(n)$ is $O(n^4)$
   (b) $f(n) - g(n)$ is $O(n^4)$
   (c) $f(n) - 2g(n)$ is $O(n^4)$
   (d) $f(n) \cdot g(n)$ is $\Theta(n^7)$
   (e) $f(n) \cdot g(n)$ is $\Theta(n^{12})$

3. Give the best asymptotic ("big-Oh") characterization of the worst case *and* the best case time complexities of the algorithm $DoAgain(A, n)$.

Algorithm *DoAgain(A,n)*
Input: Array A storing integers and of size $n > 1$.
    $sum \leftarrow 0$
    for $c \leftarrow 0$ to n^2 **do**
        if $A[0] < 0$ **then**
            for $k \leftarrow 0$ **to** $n - 1$ **do**
                $sum \leftarrow sum + c \cdot A[k]$

   (a) Best case $O(n)$ and worst case $O(n^2)$
   (b) Best case $O(n)$ and worst case $O(n^3)$
   (c) Best case $O(n^2)$ and worst case $O(n^3)$
   (d) Best case $O(n)$ and worst case $O(n^4)$
   (e) Best case $O(n^3)$ and worst case $O(n^4)$

4. Consider the algorithm below. Which set of recurrance equations characterizes its running time?

Algorithm *DoLittle(A,n)*
Input: Array A storing integers; array index n in a valid range
    if $n <= 1$
        return 0
    else
        return $A[n/3]$ + DoLittle(A,n/3)+2*DoLittle(A,n/3)

(a) $T(0) = k$
$T(n) = c + T(n/3)$

(b) $T(0) = k$
$T(n) = c + 2T(n/3)$

(c) $T(0) = k$
$T(n) = n/3 + T(n/3)$

(d) $T(0) = k$
$T(n) = c + T(2n/3)$

(e) $T(0) = k$
$T(n) = c + 3T(n/3)$

5. Suppose we create a hash table of size 19 with double hashing strategy to store positive integer keys. What is the best choice for the primary and secondary hash functions?

(a) Primary $h(k) = |k * 3 + 11| \ mod\ 19$, secondary $h'(k) = 17 - |k| \ mod\ 17$

(b) Primary $h(k) = 17 - |k| \ mod\ 17$, secondary $h'(k) = |k * 3 + 11| \ mod\ 19$

(c) Primary $h(k) = |k| \ mod\ 19$, secondary $h'(k) = 17 - |k| \ mod\ 17$

(d) Primary $h(k) = 17 - |k| \ mod\ 17$, secondary $h'(k) = |k * 3 + 11|$

(e) Primary $h(k) = |k * 3 + 11| \ mod\ 19$, secondary $h'(k) = 17 - |k|$

6. Suppose we have a hash table with N buckets, currently containing n entries. Suppose that instead of a linked list, each bucket is implemented as a binary search tree. Give the best asymptotic ("big-Oh") characterization of the worst and best case time complexity of adding an entry to this hash table.

(a) Best case $O(1)$, worst case $O(n)$

(b) Best case $O(1)$, worst case $O(\log n)$

(c) Best case $O(\log n)$, worst case $O(n)$

(d) Best case $O(n)$, worst case $O(n)$

(e) Best case $O(\log n)$, worst case $O(n^2)$

7. Let H be a heap storing 16 entries. Which of the following statements is true?

   (a) The shortest path in the heap is 2 and the longest path in the heap is 4.

   (b) The shortest path in the heap is 2 and the longest path in the heap is 3.

   (c) The shortest path in the heap is 3 and the longest path in the heap is 5.

   (d) The shortest path in the heap is 3 and the longest path in the heap is 4.

   (e) The shortest path in the heap is 3 and the longest path in the heap is 3.

8. Let H be a heap of size more than 1000 storing unique integer keys. Which of the following statements is false?

   (a) The second smallest key element must be at level 1.

   (b) The third smallest key element must be at level 2.

   (c) The forth smallest key element could be at level 2.

   (d) The fifth smallest key element could be at level 3.

   (e) The sixth smallest key element could be at level 4.

9. Let T be a full binary tree with 2011 nodes. How many internal nodes does T have?

   (a) 1004

   (b) 1005

   (c) 1006

   (d) 1007

   (e) 1008

10. Let T be a full binary tree with root r where each node stores an integer key. Consider the algorithm below. What does the algorithm return when called on the root r? Here $v.left$ and $v.right$ give the left and the right children of v, respectively, and $v.key$ is the key stored at node v.

```
Algorithm TreeAlg(v)
    if v is external
        return v.key
    else
        return TreeAlg(v.left)+TreeAlg(v.right) − 1
```

   (a) (sum of all keys at internal nodes) - (sum of all keys at external nodes)

   (b) (sum of all keys at external nodes)-(sum of all keys at internal nodes)

   (c) (number of internal nodes) - (sum of all keys at external nodes)

   (d) (sum of all keys at external nodes)-(number of internal nodes)

   (e) (number of external nodes)-(number of internal nodes)

11. Let T be an AVL tree of height 5. What is the smallest number of entries it can store? Remember that the leafs do not store any entries.

    (a) 9

    (b) 10

    (c) 11

    (d) 12

    (e) 13

12. Let T be an AVL tree of height 10. What is the largest number of entries it can store? Remember that the leafs do not store any entries.

    (a) $2^{10} - 1$

    (b) $2^9 - 1$

    (c) $2^{11} + 1$

    (d) $2^9 + 1$

    (e) $2^{11} - 1$

Part 2: Written Answers

Instructions: *Write your answers directly in these sheets. Show all the work. Use the back of the sheets if necessary.*

13. [9 marks] Prove that $f(n) = 2210 + 30n^5\ log(n) + 2n^2 + 10n$ is $O(n^5 \log(n))$ using the definition of "big-Oh".

    Solution:

    For $n_0 \geq 2$, we have

    $$f(n) = 2210+30n^5\ log(n)+2n^2+10n \leq 2210n^5 \log(n)+30n^5\ log(n)+2n^5 \log(n)+10n^5 \log(n)$$

    $$\leq (2210 + 30 + 2 + 10)n^5 \log(n).$$

    Take $n_0 = 2$ and $C = 2252$. We have that $2210 + 30n^5\ log(n) + 2n^2 + 10n \leq Cn^5\ log(n)$ for any $n \geq C$, as needed.

14. [10 marks] Consider a hash table of size 11 storing entries with integer keys. Suppose the hash function is $h(k) = k \bmod 11$. Insert, in the given order, entries with keys 0, 1, 6, 7, 10, 22, 21 into the hash table using:

(a) [4 marks] Linear probing to resolve collisions. Show all the work.

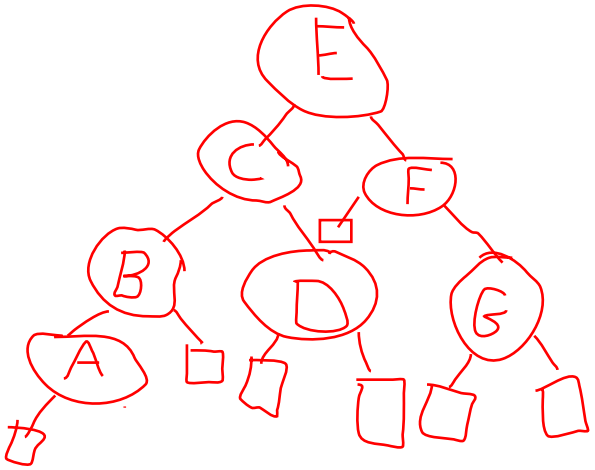
SOLUTION: (Note that you had to show the work, I'm not showing my work.)

0	1	2	3	4	5	6	7	8	9	10
0	1	22	21			6	7			10

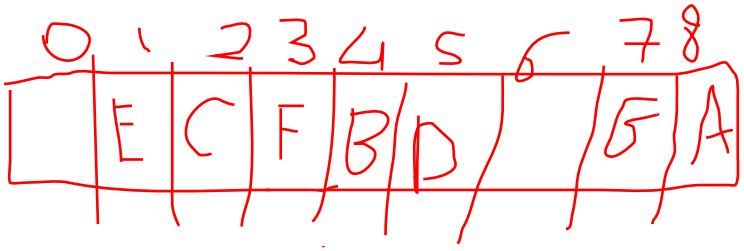
(b) [6 marks] Double hashing to resolve collisions with secondary hash function $h'(k) = 5 - (k \bmod 5)$. Show all the work.

0	1	2	3	4	5	6	7	8	9	10
0	1		22	21		6	7			10

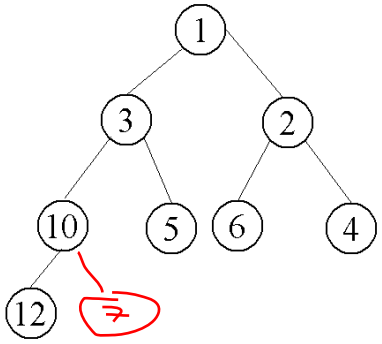
15. [10 marks].
(a) [6 marks] Draw an AVL Tree tree containing keys A, B, C, D, E, F, G such that a pre-order traversal visits nodes in order "E,C,B,A,D,F,G".

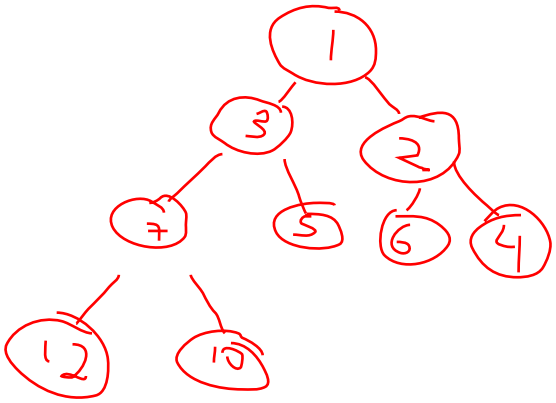


(b) [4 marks] Give the array representation of the tree you got in part (a).

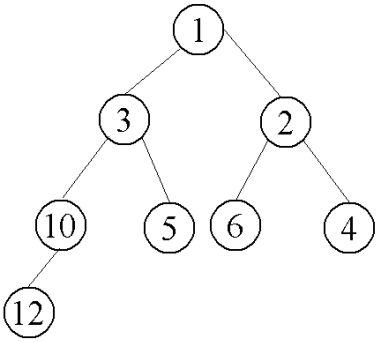


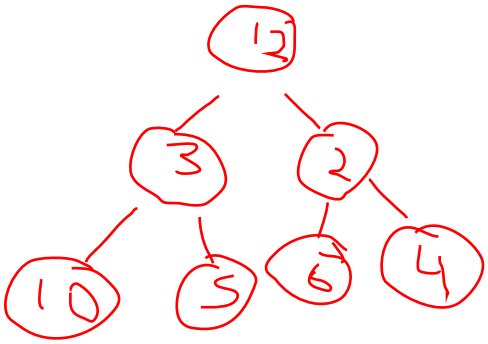
16. [5 marks] Consider the following heap. Perform operation insert(7). You have to use exactly the same algorithm as in class. Show all the intermediate trees.

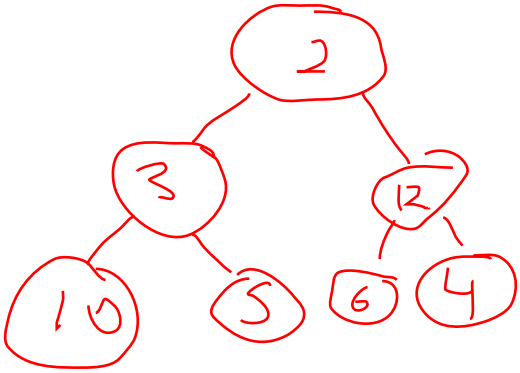


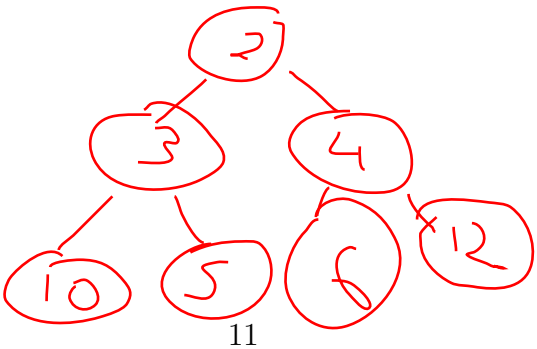


17. [5 marks] Consider the following heap. Perform operation deleteMin(). You have to use exactly the same algorithm as in class. Show all the intermediate trees.









18. [13 marks]

(a) [8 marks] Write, in pseudocode, an algorithm $Check(A, B, n)$ that takes as input arrays A and B of size n. Array A stores integers, array B is used for output. Your algorithm should modify array B in the following way. For each index i from $0 \leq i \leq n$, $B[i]$ should store the number of positive integers in A strictly to the left of i (i.e. at indexes $0, 1, ..., i-1$) minus the number of positive integers in A strictly to the right of i (i.e. at indexes $i+1, i+2, ...n-1$. For example, if $A = \{5, -4, 7, -9\}$ then the output array $B = \{-1, 0, 1, 2\}$. You are allowed to use only O(1) of additional memory. This means that is you can use a few variables, but you cannot declare arrays or any other data structures whose size depends on n. Bonus mark (+3 marks) if your algorithm has time complexity $O(n)$, but only if it is correct.

```
Alg Check (A,B,n)
for i = 0 to n-1                          n
    numLeft = 0                           n
    for j = 0 to i-1                      n^2
       if A[j] > 0                        n^2
           numLeft++                      n^2

    numRight = 0                          n
    for j = i+1 to n                      n^2
       if A[j] > 0                        n^2
          numRight++                      n^2
     B[i] =  numLeft - numRight
```

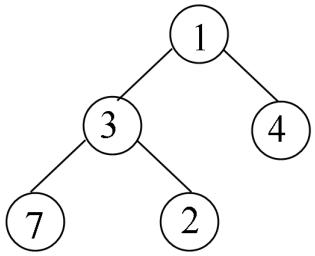
```
Alg CheckLInear(A,B,n)
totalPos = 0                              1
for i = 0 to n-1                          n
    if A[i] > 0                           n
        totalPos++                        n
numLeft = 0                               1
numRight = totalPos                       1
for i = 0 to n-1                          n
    if A[i] > 0                           n
        numRight--                        n
     B[i] = numLeft-numRight              n
     if A[i] > 0                          n
       numLeft++                          n
```

(b) [5 marks] Give the best "big-Oh" characterization of the worst case time complexity of your algorithm above. Express the complexity as a function of the array size n. Explain how you computed the time complexity of your algorithm.

complexities are as noted above

19. [12 marks]

(a) [8 marks] Let T be a binary tree where each node stores an integer key that you can access with $v.key$. Write a recursive algorithm $Alg(r)$ that receives as input the root of the tree r and outputs the sum over the node key multiplied by the depth. For example, for the tree in the picture below, your algorithm should return $1 \cdot 0 + 3 \cdot 1 + 4 \cdot 1 + 7 \cdot 2 + 2 \cdot 2 = 25$ since keys $1, 3, 4, 7, 2$ are at depths, respectively, $0, 1, 1, 2, 2$.



Solution 1

```
Alg(r)
    return AlgMain(r,0)
```

```
AlgMain(v,d)
    if isExternal(v)                          1
        return(v.key*d)                       1

    if v.left != null                         1
       rLeft = AlgMain(v.left,d+1)            1

    if v.right != null                        1
        rRight = AlgMain(v.right,d+1)         1
   return(rLeft+rRight+v.key*d)               1
```

Solution 2

```
Alg(v)

depth = 0                                     1
temp = v                                      1
while temp.parent != null                     n
          depth++                             n
          temp = temp.parent                  n
if isExternal(v)                              1
      return(depth*v.key)                     1
else if v.left != null                        1
        rLeft = Alg(v.left)

    if v.right != null                        1
        rRight = Alg(v.right)
   return(rLeft+rRight+v.key*depth)           1
```

(b) [4 marks] Give the best "big-Oh" characterization of the time complexity of your algorithm above. Express the complexity as a function n, the number of nodes. Explain how you computed the time complexity of the algorithm.

The Solution 1 algorithm above visits each node one time, and at each visitation, it performs a constant number of operations. Since number of nodes is n, total time C*n which is O(n)

Solution 2:
The algorithm spends O(n) time at each node, and visits each node exactly one time. Therefore the total time is O(n^2)