CMSC132 Summer 2017 Midterm 2

First Name (PRINT): _____

Last Name (PRINT): _____

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

Instructions

- This exam is a closed-book and closed-notes exam.
- Total point value is 100 points.
- The exam is a 80 minutes exam.
- Please use a pencil to complete the exam.
- WRITE NEATLY. If we cannot understand your answer, we will not grade it (i.e., 0 credit).

1. Multiple Choice	/ 25
2. Short Answer	/ 45
3. Programming	/ 30
Total	/100

1. [25 pts] Multiple Choice

Identify the choice that best completes the statement or answers the question. **Circle your answer.**

1) [2] The cost of inserting or removing an element to/from a heap is log(N), where N is the total number of elements in the heap. The reason for that is:
   a) Heaps keep their entries sorted
   b) Heaps are balanced BST.
   c) Heaps are balanced binary trees.
   d) Heaps are a version of Red-Black trees

2) [2] What is the output of **`fun(2)?`**

```
int fun(int n){
  if (n == 4)
    return n;
  else
    return 2*fun(n+1);
}
```

   a) 4
   b) 8
   c) 16
   d) Runtime Error

3) [2] 2-3-4 Tree guarantees searching in _____time.
   a) O(log n)
   b) O (n)
   c) O (1)
   d) O (n log n)

4) [2] What is the advantage of an iterative method over a recursive one?

   a. It makes fewer calls
   b. It has less overhead
   c. It is easier to write, since you can use the method within itself.
   d. It uses a stack

5) [2] A priority queue can be efficiently implemented using which of the following data structures?

   a) Array
   b) Linked List
   c) Binary Heaps
   d) None of the above

6) [2] A Priority-Queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is given below: 10, 8, 5, 3, 2. What is the level order traversal of the heap after inserting 1 and 7.

   a) 10, 8, 7, 5, 3, 2, 1
   b) 10, 8, 7, 2, 3, 1, 5
   c) 10, 8, 7, 1, 2, 3, 5
   d) 10, 8, 7, 3, 2, 1, 5

7) [2] What is the worst-case time complexity for insert and remove operations in a Binary Heap?

   a) O(n) for all
   b) O(log n) for all
   c) O(log n) insert, and O(n) for remove
   d) O(log n) for remove, and O(n) for insert

8) [2] In an array based implementation (root index is 1) of a Binary Heap, the children of a node at position k are at positions:

   a) 2k and 2k-1
   b) 2k+1 and 2k+2
   c) 2k and 2k+1
   d) 2(k+1) and 2(k+1)+1

9) [3] The following numbers are inserted into an empty binary search tree in the given order: 10, 12, 3, 5, 14, 2, 11, 18. What is the height of the binary search tree (the height is the maximum distance (number of edges) of a leaf node from the root, height of a tree with one node is 0.)?

   a) 5
   b) 3
   c) 4
   d) 2

10) [3] Here is an INCORRECT pseudo code for the algorithm which is supposed to determine whether a sequence of parentheses is balanced:

```
declare a character stack
while ( more input is available) {
   read a character
   if (the character is a '(' )
        push it on the stack
    else if( the character is a ')' and the stack is not empty )
        pop a character off the stack
    else
        print "unbalanced" and exit
}
print "balanced"
```

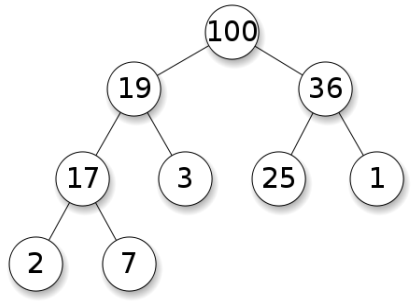
Which of these unbalanced sequences does the above code think is balanced?

a) ((())
b) (()()))
c) ())(()
d) ((()))()

11) [2] What is the complexity of 2-3-4 Tree insert/delete/search operations?

a) O(log n)
b) (n)
c) (1)
d) (n log n)

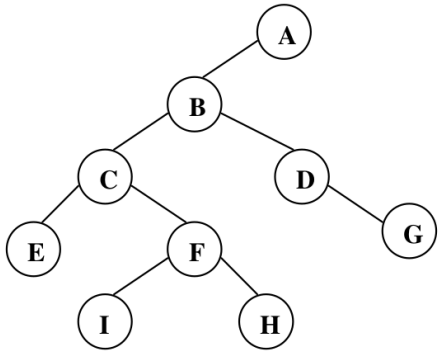
12) What is the number of swaps that occur after inserting 20 into the following heap:



a) 0
b) 1
c) 2
d) 3

2. [45 pts] Short Answer

1) [6] Write the preorder, inorder, and postorder traversal of the following binary tree:



preOrder	
inOrder	
postOrder	

2) [3] What does this function do?

```
int foo(Node n) {
    if (n == null) return 0;
    if (n.left == null && n.right == null) return 1;
    return foo(n.left) + foo(n.right);
}
```

Answer:

3) [3] Suppose that you do **binary search** for the key 39 in the following sorted array of size 15:

10 11 25 31 36 39 53 55 56 64 68 75 78 82 87

Give the sequence of keys in the array that are compared with 39 Answer:

4) [3] If the given preOrder traversal for a **binary search tree** is {10, 3,1, 7, 15, 20, 25}, construct the binary search tree.

5) [4] Given the following contents of an array implementation of a stack, where 50 is at the top of the stack.

0	1	2	3	4	5
20	30	40	50		

Show the contents of the stack and the location of top after doing the following

```
stack.pop();
stack.push(60);
stack.push(70);
stack.push(80);
stack.pop();
stack.push(90)
```

0	1	2	3	4	5

6) [4] Given the following contents of a circular array implementation of a queue

0	1	2	3	4	5
20	30	40			

first last

Show the contents of the queue and locations of first and last after doing the following:

```
Queue.dequeue();
Queue.dequeue();
Queue.enqueue(50);
Queue.enqueue(60);
Queue.enqueue(70);
Queue.dequeue();
Queue.enqueue(80);
```

0	1	2	3	4	5

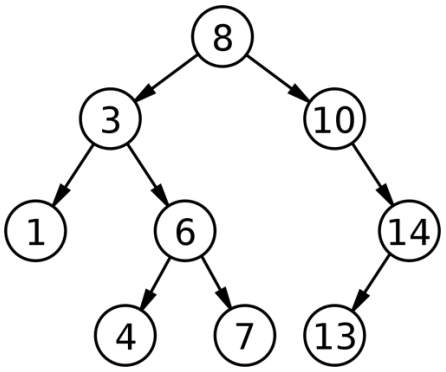
7) [4] Draw the binary heap shown in the form of an array.

0	1	2	3	4	5	6	7	8	9	10
--	16	14	10	8	7	9	3	2	4	1

8) [4] Show the contents of the heap after the value 15 is inserted.

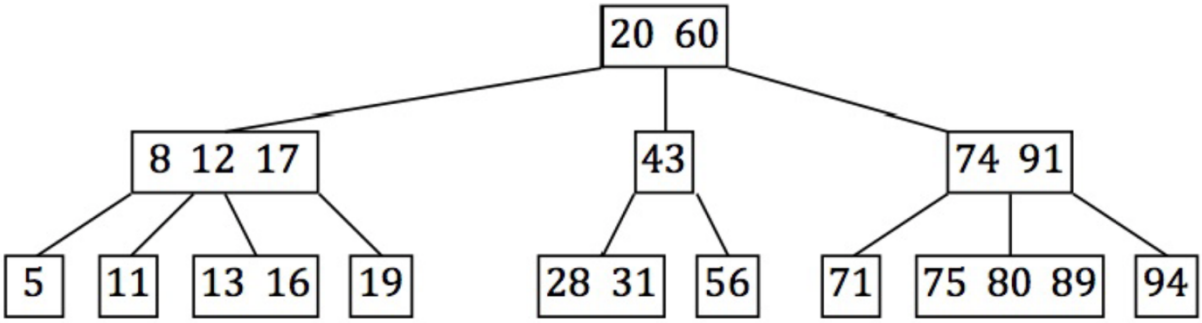
0	1	2	3	4	5	6	7	8	9	10	11
--											

9) [3] Draw the Binary Search Tree after you delete key 8.



10) [5]Construct a left-leaning red-black tree when the following elements are inserted in this order: 20, 10, 15, 18, 30. Show your steps for each number. Use dashed line for red edge.

11) Given a 2-3-4 tree as follows



a) [3] Draw the 2-3-4 after inserting 44, 84,86 into this tree

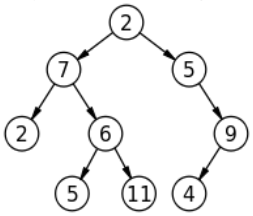
b) [3] Draw the tree after deleting 94 from the original 2-3-4 tree

3. [30 pts] Programming questions

Use the following Node definition for questions 1)—4)

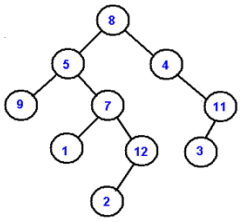
```
class Node{
      int key;
      Node left, right;
}
```

1) [6] Write a method "**`int sum(Node r)`**", which returns the sum of all keys in a given binary tree. Calling sum on the following tree will return 51. Sum of an empty tree is 0.

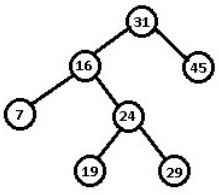


2) [8] Write a method "**`boolean find(Node r, int key)`**", which receives a Binary Search Tree r and a key as arguments, and returns if the key exists in the tree

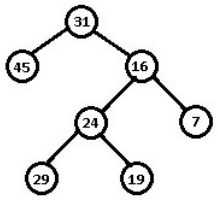
3) [8] Write a method "`int getMin(Node r)`", which returns the minimum key in the given **Non-Empty** Binary Tree. Calling getMin on the following tree returns 1.



4) [8] write the method "`Node mirror(Node r)`", which returns the mirror of the given binary tree. For example, calling mirror on the original tree, will return the mirror tree shown below.



Original Tree



Mirror Tree