Midterm 1

CS 14 - Data Structures

May 13, 2013

By taking this exam, I affirm that all work is entirely my own. I have not cheated in any way.

Signature: _____

Printed Name: _____

1. (3 pt) What does $f = \Theta(g)$ mean? Give an exact definition.

2. (2 pt) What is the difference between a data structure and an ADT?

3. (5 pt) State the solutions to these recurrence relations using Θ notation. There is no need to show work.

   (a) $T(n) = T\left(\frac{n}{2}\right) + 1$; $T(1) = 1$

   (b) $T(n) = 2T\left(\frac{n}{2}\right) + 1$; $T(1) = 1$

   (c) $T(n) = 2T\left(\frac{n}{2}\right) + n$; $T(1) = 1$

   (d) $T(n) = T(n-1) + 1$; $T(1) = 1$

   (e) $T(n) = T(n-1) + n$; $T(1) = 1$

4. (2 pt) Solve the bitwise equations. All numbers are in binary.

   (a) 10100011 ^ 00001111

   (b) ~11110011

5. (4 pt) List four example uses of the Map ADT. You must explicitly state what the keys and values are.

   (a)

   (b)

   (c)

   (d)

6. (3 pt) List three example uses of the Set or Multiset ADT.

   (a)

   (b)

   (c)

7. (3 pt) List three example uses of the Priority Queue ADT.

   (a)

   (b)

   (c)

8. (4 pt) State precisely the invariants maintained by an AVL tree. Use pseudocode if needed.

9. (4 pt) State precisely the invariants maintained by a binary heap. Use pseudocode if needed.

10. (3 pt) Define the load factor. State which data structure it is used for and the consequences of a high and low load factor.

11. (3 pt) Define a *perfect hash function* a describe when you can use it.

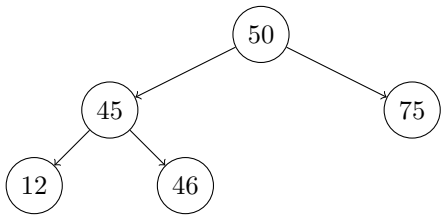
12. (10 pt) Fill out the table of run times for each function.

data structure	function	best case run time	worst case run time
binary search tree	insert		
	delete		
	search		
AVL tree	insert		
	delete		
	search		
hash table separate chaining (linked lists)	insert		
	delete		
	search		
hash table separate chaining (AVL trees)	insert		
	delete		
	search		
hash table open addressing	insert		
	delete		
	search		
binary heap	insert		
	delete		
	find min/max		

13. (4 pt) Draw the **binary search tree** structure that is created from the following code.

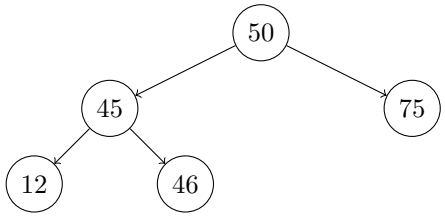
```
BST t; // creates an empty BST
for (int i=0; i<5; i++)
    t->insert(i);
```

14. (4 pt) Given this **binary search tree**:



Draw the resulting tree after deleting the number 75.

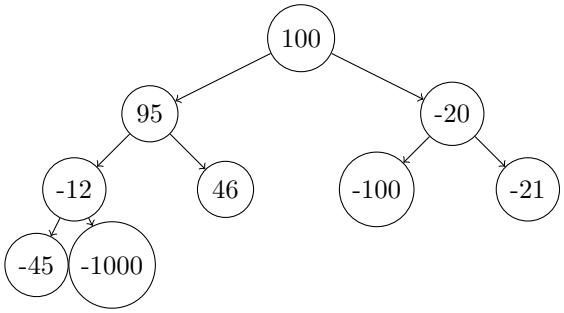
15. (4 pt) Given this **AVL Tree**:



Draw the resulting tree after inserting the number 47.

16. (4 pt) Using the **original** AVL tree structure before inserting, draw the resulting tree after deleting the number 50.

17. (4 pt) Given this **Binary Heap**:



Draw the resulting tree after inserting the number 47.

18. (4 pt) Using the **original** binary heap structure before inserting, draw the resulting tree after deleting the number -20.

19. (10 pt) Given this code for a tree structure:

```
template<typename T> struct Tree
{
    T val;
    Tree<T> *left, *right;
}
```

Write a search function that returns true if searchVal is contained within the tree, false otherwise. You must use a breadth first search. You may not make any assumptions about the tree's structure. For example, the tree might not obey the binary search tree property.

```
template<typename T>
bool breadthFirstSearch(Tree<T> *root, T searchVal)
{









}
```

20. (10 pt) The following binary heap class has been implemented. It stores the smallest values on top.

```
class BinHeap
{
    int val;

public:
    BinHeap();
    void insert(int);
    void delete(int);
    int deleteMin();
    int peekMin();
}
```

Use this class to implement a heap sort function. The variable v is *both* the output and the input to your function. When it is output, it must be sorted from highest to lowest. That is, v[0] should be the largest element in the vector.

```
void heapSort(vector<int> &v)
{




}
```

21. (10 pt) Given the AVL tree structure:

```
template <typename V>
struct AVLTree
{
    V val;
    AVLTree<V> *left, *right;
}
```

Write a function that returns the height of the tree. You must use the provided function prototype.

```
template <typename V>
int height(AVLTree<V> *t)
{

}
```

Extra credit

22. (2 pt) Name a binary search tree besides AVL trees that is guaranteed to be balanced.

23. (2 pt) What does AVL stand for? (You don't have to know exactly, just the general idea.)

24. (2 pt) Google, Microsoft, IBM, and every major company have specific rules their programmers must follow. These rules cover how to format code, how to comment, which libraries to ues, etc. What are these rules called?

25. (2 pt) What is your TA's name?