Computer Science 302 Spring 2007 Practice Final Examination: Part I

Name:____

This practice examination is much longer than the real final examination will be. If you can work all the problems here, you will probably be well-prepared for the final examination.

This part of the practice examination is 335 points.

1. True or False. [5 points each]
   (a) ________ The time to heapsort an array of n items is $\Theta(n \log n)$.
   (b) ________ Open hashing uses open addressing.
   (c) ________ In the decision tree model of computation, the time complexity of any algorithm to sort n items is $\Omega(n \log n)$.
   (d) ________ The height of a binary tree with n nodes is $\Omega(\log n)$.

2. [10 points] What implementation of the ADT search structure would you use if the expected number of items in the structure is 1?

3. [10 points] What implementation of the ADT search structure would you use if n items are to be inserted at once at the beginning of the program, there will be no further inserts, and find will be executed n^2 times during the running of the program? (There is more than one correct answer to this problem, as well as several inferior answers.)

4. [10 points] Suppose an $5 \times 6 \times 8$ array A is stored in column-major order, with base address 1000, and one word is required for each entry. What is the address of A[3,2,6]? Assume that array indices start at 0, as in C++.

5. [20 points] Write pseudocode for the array implementation of the ADT "stack of integer." Your code should include procedures that implement **pop**, **push**, and **empty**.

6. [20 points] Explain how you would insert and delete from a queue, given that you are using singly linked nodes in a circular linked list implementation.

7. [30 points] Describe each of the following types of search. (Be sure to say what the structure is that is being searched in each case.)

   (a) Linear search.

   (b) Binary search.

8. [40 points] Give four ways to implement the ADT ”search structure.” (For example, ”binary search tree” is one way, so don't use that one.) Just give the names of the implementations, not any details.

9. [10 points] You can sort a set of items using a binary search tree, as follows: Start with an empty binary search tree, insert the items one at a time, then visit the nodes of the tree in inorder, writing out the items. This algorithm is essentially the same as which one of these well-known sorting algorithms? (Choose one answer.)

   (a) Quicksort

   (b) Heapsort

   (c) Mergesort

10. [10 points each] For each of the following code fragments, express the asymptotic time complexity by choosing the best of the following answers: $O(n)$, $O(n^2)$, $O(n \log n)$, $O(\log n)$, $O(\log \log n)$, $\Theta(n)$, $\Theta(n^2)$, $\Theta(n \log n)$, $\Theta(\log n)$, $\Theta(\log \log n)$

   (a)
```
for (int i = 0; i < n; i++)
  cout << "Hi there.";
```

   (b)
```
for (int i = 0; i < n; i = 2*i+1)
  cout << "Hi there.";
```

(c)
```
for (int i = 0; i < 0; i++)
  for (int j = i; j > 0; j = j/2);
  cout << "Hi there.";
```

(d)
```
for (int i = 0; i < n; i++)
  for (int j = i; j > i/2; j = j/2);
    cout << "Hi there.";
```

(e)
```
for (int i = 0; i < n; i = i*i+1)
   cout << "Hi there.";
```

(f)
```
   for (int i = 0; i < n; i++)
    {
     int j = unknown(i);
// unknown is a function whose value could be anything: we have no clue!
     if (i < j)
       i = j;
     cout << "Hi there.";
    }
```

11. [30 points]

(a) Describe the meaning of the word collision as used in discussions of hashing.

(b) How are collisions handled in closed hashing?

(c) How are collisions handled in open hashing?

12. [25 points] Explain how insertion works in a B-tree. Hint: the phrase "node splitting" or the equivalent must be in your explanation.

13. [20 points] Write C++ code for the **find** portion of union-find as you used in Assignment 10. Be sure to copy the code that you wrote for that assignment as precisely as you can, using the same variable names. (I hope you used path compression as you were supposed to.) Do not include any other part of the program. If you write more than 10 lines, you've written far too much.

14. [30 points] Given the following:

```
class BST{ // Binary Search Tree
  public:
   BST(int); // initializes item field to parameter, links to 0
   void static inorderWrite(BST*); // all items to standard ostream inorder
   void static insert(int, BST*&); // inserts parameter, if not there
   bool find(int, BST*); // parameter is in the binary search tree
  private:
   int item; // the value stored in the node
   BST * left; // pointer to the left subtree
   BST * right; // pointer to the right subtree
 };
```

Complete the following code by writing **exactly three lines:**

```
     void BST::inorderWrite(BST * t){
        if (t != 0){
// Your three lines go here.




          }
      }
```

335. Points.