Student ID: _____

Graduate Comprehensive Exam: Data Structures and Algorithms (Spring 2002)

Answer all questions on the exam. You may use the back for additional space. Total: 100 points. Good Luck.

1. (20 pts) The following states two tasks and four data structures for each task. Each data structure contains N unique numbers.

   (a) checking if x is one of the N numbers

      i. a sorted array
      ii. a balanced binary search tree
      iii. a hash table with a perfect hash function
      iv. a heap

   (b) finding the largest number:

      i. a sorted array
      ii. a balanced binary search tree
      iii. a hash table with a perfect hash function
      iv. a heap

   Briefly state/name your algorithm to accomplish the two tasks using each of the four data structures and *discuss* its worst-case time complexity in big-O (only counting the number of comparisions between x and an element in a data structure or between two elements in a data structure). That is, 8 algorithms (briefly) and 8 worst-case analyses.

2. (50 pts) C, C++, Java, or pseudocode with **sufficient** details can be used for this two-part question:

   (a) (25 pts) Write a recursive function that evaluates a polynomial $a_0 + a_1x + a_2x^2 + ... + a_nx^n$ where the $n + 1$ coefficients a_i are passed to the function in an array along with the degree n. The function has 3 parameters: the coefficients in an array a, variable x, and degree n. Also, you may assume a, x, and n are `int`'s.

   (b) (25 pts) Two trees are *isomorphic* if they have the same *structure* independent of the the values stored in each node. Write a function that compares two binary trees and tells whether they are isomorphic. If you want, you may assume that a reference for both trees are passed to the function.

3. (10 pts) Using the big-O notation, estimate the running time of `sum(numArray, 0, N-1)` in terms of `N` which is a positive integer. For this estimation of running time, you may consider only counting the number of additions in "`line add`" as marked in the comment below. Explain your answer.

```
int sum(int a[], int start, int end)
{
  if (start < end)
    {
      int mid = (start + end) / 2;
      return sum(a, start, mid) + sum(a, mid+1, end);  // line add
    }
  else if (start == end)
    return a[start];
  else
    return 0;
}
```

4. (20 pts) Sorting

   (a) Given an array of these integers: 4 9 7 3 2 2 8, peform Insertion Sort and show your steps. Show the array after each pass of the array (after one more item is sorted).

   (b) For counting the number of comparisons in Mergesort, when does the worst case occur and what is the complexity in big-O (explain)?