Carleton UNIVERSITY

FINAL
EXAMINATION
Fall 2004

<u>DURATION: 3 HOURS</u> **No. Of Students: 20**

Department Name & Course Number: Systems and Computer Engineering SYSC 2100

Course Instructor (s): Thomas Kunz

AUTHORIZED MEMORANDA

NONE

Students MUST count the number of pages in this examination question paper before beginning to write, and report any discrepancy to a proctor. This question paper has _____7_____pages + cover page = __8_ pages in all.

This examination question paper may not be taken from the examination room.

In addition to this question paper, students require: an examination booklet no
Scantron Sheet no

Name: ____

Student Number: ____

Question 1: Algorithm Analysis (10 marks)

Four "Mystery" classes have been created that implement various collections. Using different values for n, sample run-times were measured (similar to assignment 1). With these, hypothesize what the time estimate (log n, quadratic, etc.) is for each Mystery class:

Mystery1.class

n Value	Run-Time
n = 30	1.797 seconds
n = 60	7.204 seconds

Mystery2.class

n Value	Run-Time
n = 250	0.061 seconds
n = 500	0.061 seconds

Mystery3.class

n Value	Run-Time
n = 10000000	0.485 seconds
n = 20000000	1.638 seconds

Mystery4.class

n Value	Run-Time
n = 8	0.0010 seconds
n = 16	0.088 seconds

Question 2: ADT List (10 marks)

The Java Collections Framework provides two implementations of the ADT List: *ArrayList* and *LinkedList*.

1. Explain why there are two distinct implementations of that ADT
2. Briefly sketch the code for the following task (task1): for each of *n* indexes, randomly generated, retrieve the list element at that index.
3. Briefly sketch the code for the following task (task2): repeatedly remove the first element (at index 0) from a list until the list is empty
4. Hypothesize the runtime complexity of each piece of code when using ArrayList and when using LinkedList.

Question 3: Bags, Stacks, and Queues (10 marks)

1. Suppose that rateSet is a Bag object of Double elements. Write the code to print each element in rateSet whose value is greater than 0.5.
2. Suppose we added each of the following methods to the ArrayList class:

   public boolean addFirst (E element)
   public boolean addLast (E element)
   public E getFirst()
   public E getLast()
   public E removeFirst()
   public E removeLast()

   Estimate worstTime(*n*) for each method.

Question 4: Binary Trees/Binary Search Trees (15 marks)

1. Define the *depth()* method for a binary tree. Assume that the internal representation of the tree is based on a the node structure we discussed in class:

   ```
   private static class BTNode {
       Object element;
       BTNode left;
       BTNode right;
   }
   ```

   and that the class contains an instance variable root that contains the root of the tree.

   ```
   /**
    * The depth of this BinarySearchTree has been calculated and returned.
    *
    * @return an int containing the height of the BinarySearchTree.
    **/
   public int depth();
   ```

2. Describe in English how to remove each of the following from a binary search tree:

   a. an element with no children

   b. an element with one child

   c. an element with two children

Question 5: Sorting (15 marks)

1. InsertionSort is another way to sort a collection, besides HeapSort. Assume we want to sort an array of integers. The following code implements insertionSort:

```
public static void insertionSort (int[ ] x)
{
      for (int i = 1; i < x.length; i++)
            for (int j = i; j > 0 && x [j -1] > x [j]; j--)
                  swap(x[i], x[j]); // exchanges the values
} // method insertionSort
```

   Estimate the runtime complexity (averageTime(n) and worstTime(n)) for InsertionSort.

2. Consider the following, consecutive improvements to Insertion Sort:

   a) Replace the call to the method swap with in-line code:

```
public static void insertionSort (int[ ] x)
{
      int temp;
      for (int i = 1; i < x.length; i++)
            for (int j = i; j > 0 && x [j -1] > x [j]; j--)
            {
                  temp = x [j];
                  x [j] = x [j - 1];
                  x [j - 1] = temp;
            } // inner for
} // method insertionSort
```

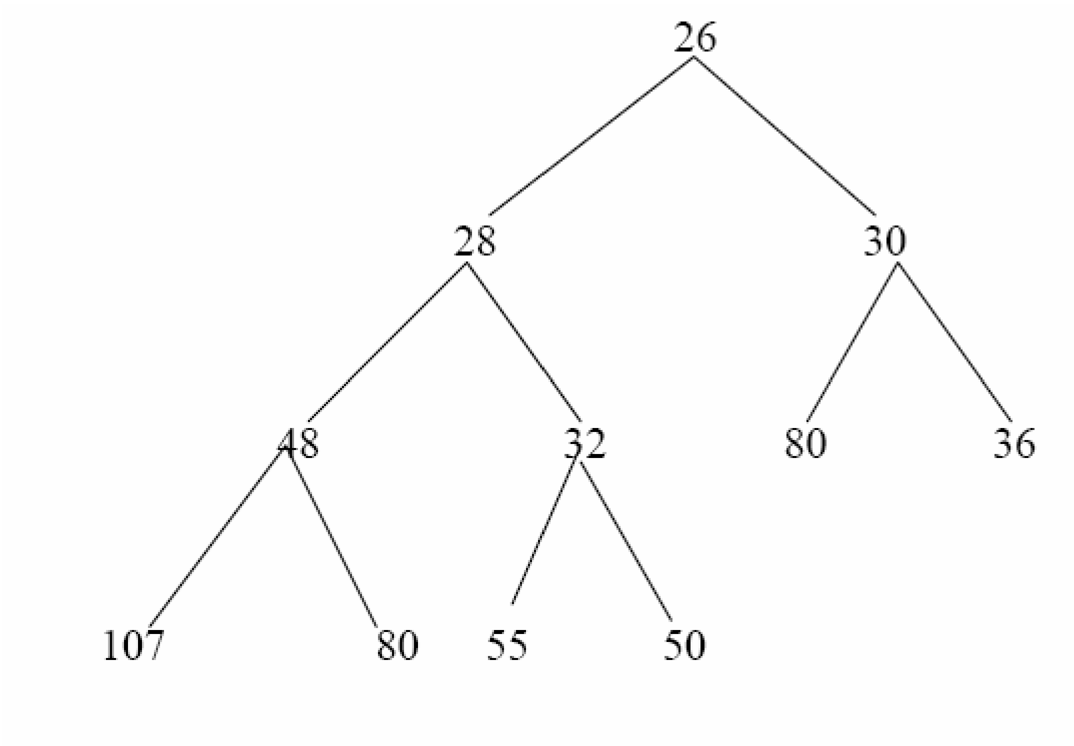
   b) Notice that in the inner loop in part a), temp is repeatedly assigned the original value of x [i]. For example, suppose the array x has 32 46 59 80 35 and j starts at 4. Then 35 hops its way down the array, from index 4 to index 1. The only relevant assignment from temp is that last one. Instead, we can move the assignments to and from temp out of the inner loop:

```
int temp, j;
for (int i = 1; i < x.length; i++)
{
      temp = x [i];
      for (j = i; j > 0 && x [j -1] > temp; j--)
            x [j] = x [j - 1];
            x [j] = temp;
} // outer for
```

   Will these changes affect the estimates for worstTime(n) and averageTime(n)?

Question 6: Heaps/Priority Queues (15 marks)

1. Show the resulting heap after each of the following alterations is made, consecutively, to the following heap:



    - add (29)
    - add (30)
    - removeMin(); removeMin()
2. If each of the letters 'a' through 'f' appears at least once in the original message, explain why the following cannot be a Huffman code:
   a: 1100 b: 11010 c: 1111 d: 1110 e: 10 f: 0
3. For the following character frequencies, create the heap of character-frequency pairs (highest priority = lowest frequency). Assume that the pairs are added in alphabetical order to an initially empty heap.
   a: 5,000
   b: 2,000
   c: 10,000
   d: 8,000
   e: 22,000
   f: 49,000
   g: 4,000

Question 7: Hashing (15 marks)

1. Assume that p is a prime number. Use modular algebra to show that for any positive integers *index* and *offset* (with offset not a multiple of p), the following set has exactly p elements:

   $\{ (index + k * offset) \% p;\ k = 0, 1, 2, \ldots, p - 1\}$
2. Compare the space requirements for chained hashing and open-address hashing. Assume that a reference occupies four bytes and a boolean value occupies one byte. Under what circumstances (size, loadFactor, table.length) will chained hashing require more space? Under what circumstances will double hashing require more space?
3. In open-addressing with double hashing, insert the following keys into a table of size 13:

   20, 33, 49, 22, 26, 202, 140, 508, 9

   The second hash function calculates the quotient of the key and the table size, also called open-addressing with quotient-offset collision handler.

   Here are the relevant remainders and quotients:

Key	**key % 13**	**key / 13**
20	7	1
33	7	2
49	10	3
22	9	1
26	0	2
202	7	15
140	10	10
508	1	39
9	9	0