SYSC 2100, Fall 2005 Midterm
November 1, 2005
Duration: 80 minutes
Instructor: T. Kunz

Name: **Student number:**

Question 1: Algorithmic Complexity (10 marks)

a. Suppose we have a method whose worstTime(n) is linear in n. Determine the effect of tripling n on the estimate of worst time.That is, estimate worstTime($3n$) in terms of worstTime(n).

b. Suppose we have a method whose worstTime(n) is quadratic in n. Determine the effect of tripling n on the estimate of worst time. That is, estimate worstTime($3n$) in terms of worstTime(n).

c. Suppose we have a method whose worstTime(n) is constant. Determine the effect of tripling n on the estimate of worst time. That is, estimate worstTime($3n$) in terms of worstTime(n).

Question 2: Recursion (10 marks)

A ***permutation*** is an arrangement of elements in a linear order. For example, if the elements are the letters 'A', 'B', 'C' and 'D', we can generate the following 24 permutations:

ABCD BACD CABD DABC ABDC BADC CADB DACB ACBD BCAD CBAD DBAC
ACDB BCDA CBDA DBCA ADBC BDAC CDAB DCAB ADCB BDCA CDBA DCBA

Perform an execution-frames trace to determine the output from the following ***(potentially incorrect)*** version of the recPermute method after an initial call to permute ("ABC") invokes recPermute (['A', 'B', 'C'], 0);

```
/**
* Finds all permutations of a subarray from a given position to the end of
* the array.
*
* @param c an array of characters
* @param k the starting position in c of the subarray to be permuted.
*
* @return a String representation of all the permutations.
*
*/
public static String recPermute (char[ ] c, int k)
{
        if (k == c.length - 1)
                return String.valueOf (c) + "\n";
        else
        {
                String allPermutations = new String();
                char temp;
                for (int i = k; i < c.length; i++)
                {
                        allPermutations += recPermute (String.valueOf (c).toCharArray(), k+1);
                        temp = c [i];
                        c [i] = c [k];
                        c [k] = temp;
                } // for
                return allPermutations;
        } // else
} // method recPermute
```

Question 3. ADT List (10 marks)

a. State two advantages, and one disadvantage, of using an ArrayList object instead of an array object.

b. Show that, for the task of appending n elements to an ArrayList object, worstTime(n) is linear in n.

c. The one-parameter add method in the ArrayList class always returns **true**. Would it make sense to change the return type from **boolean** to **void**? Explain.

d. Hypothesize the output from the following code:

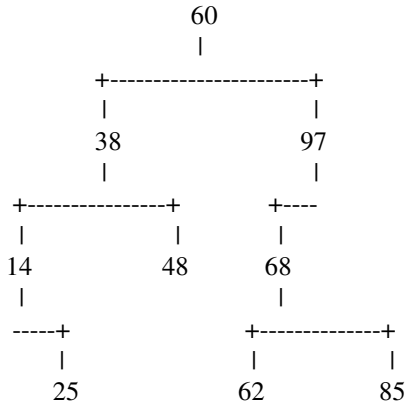
```
ArrayList letters = new ArrayList();

letters.add ("f");
letters.add (1, "i");
letters.add ("e");
letters.add (1, "r");
letters.add ("e");
letters.add (4, "z");
System.out.println (letters);

letters.remove ("i");
int index = letters.indexOf ("e");
letters.remove (index);
letters.add (2, "o");
System.out.println (letters);
```

Question 4. Binary Trees (10 marks)

a. For the following binary tree, show the order in which elements would be visited for an inOrder, postOrder, preOrder traversal.



b. Show that a binary tree with n elements has $2n + 1$ subtrees (including the entire tree). How many of these subtrees are empty?