SYSC 2100, Fall 2004 Midterm
October 28, 2004
Duration: 80 minutes
Instructor: T. Kunz

Name: **Student number:**

Question 1. Algorithm Complexity (10 marks)

For each of the following code segments, estimate worstTime(n) using Big-O notation or plain English. In each segment, S represents a sequence of statements in which there are no n-dependent loops.

```
1) for (int i = 0; i * i < n; i++)
        S
```

```
2) for (int i = 0; Math.sqrt (i) < n; i++)
        S
```

```
3) int k = 1;
   for (int i = 0; i < n; i++)
        k *= 2;
   for (int i = 0; i < k; i++)
        S
```

Question 2. The Java Collections Framework (10 marks)

1. Identify each of the following as either an interface or a class: (2 marks)
   Collection
   LinkedList
   Iterator
   AbstractSet
   Map

2. What is the difference between an interface and an abstract class? (2 marks)

3. Of what value is an abstract class? That is, to what extent can an abstract class make a programmer more productive? (3 marks)

4. What is a list? What is a set? What is a map? (3 marks)

Question 3. ADT List (10 marks)

1. Suppose we added each of the following methods to the ArrayList class:
   - **public boolean** addFirst (Object element)
   - **public boolean** addLast (Object element)
   - **public** Object getFirst()
   - **public** Object getLast()
   - **public** Object removeFirst()
   - **public** Object removeLast()

   Estimate worstTime(*n*) for each method. (2 marks)

2. In the Java Collections Framework, the LinkedList class is designed as a circular, doubly-linked list with a dummy entry (pointed to by the header field). What is the main advantage of this approach over a circular, doubly-linked list with head and tail fields? (2 marks)

3. Suppose we have the following:

```
LinkedList weights = new LinkedList();
ListIterator itr;
weights.add (new Double(5.3));
weights.add (new Double(2.8));
itr = weights.listIterator();
```

   Bad question: adding (as an iterator method) is not discussed in the course notes

   Hypothesize which of the following sequences of messages (i.e., method invocations) would now be legal: (6 marks)

   a. itr.add (new Double(8.8)); itr.next(); itr.remove();
   b. itr.add (new Double(8.8)); itr.remove(); itr.next();
   c. itr.next(); itr.add (new Double(8.8)); itr.remove();
   d. itr.next(); itr.remove(); itr.add (new Double(8.8));
   e. itr.remove(); itr.add (new Double(8.8)); itr.next();
   f. itr.remove(); itr.next(); itr.add (new Double(8.8));

Question 4. Stacks and Queues (10 marks)

1. Suppose that elements "a", "b", "c", "d", "e" are pushed, in that order, onto an initially empty stack, which is then popped four times, and as each element is popped, it is enqueued into an initially empty queue. If one element is then dequeued from the queue, what is the *next* element to be dequeued? (2 marks)

2. The array-based implementation of the Queue interface had *elementData*, *size*, *front* and *back* instance variables. Show that the *size* instance variable is redundant. Specifically, show that the *size*() method can be implemented using only the *back*, *front* and *elementData* variables. Assume that entries in *elementData* that do not contain an element are set to *null*. (3 marks)

3. An expression in postfix notation can be evaluated at run time by means of a stack. For simplicity, assume that the postfix expression consists of integer values and binary operators only. For example, we might have the following postfix expression:
   8 5 4 + * 7 -

   The evaluation proceeds as follows: When a value is encountered, it is pushed onto the stack. When an operator is encountered, the first and second elements on the stack are retrieved and popped, the operator is applied (the second element is the left operand, the first element is the right operand) and the result is pushed onto the stack. When the postfix expression has been processed, the value of that expression is the top (and only) element on the stack. For example, for the above expression, the contents of the stack would be as follows:

```
                            4
                  5         5         9                   7
        8         8         8         8         72        72        65
-----   -----     -----     -----     -----     -----     -----     -----
```

   Convert the following expression into postfix notation and then use a stack to evaluate the expression:
   5 + 2 * (30 - 10 / 5) (5 marks)