Honors Introduction To Computer Science II

CS 114H

Midterm Exam

Fall 2012

October 29, 2012

Do not open this exam until instructed to do so. The exam consists of 5 problems. Please check that you have all the pages.

The answers to the exam problems should be written in the space provided with the question. Read each question carefully before answering. Make sure you print your answer **neatly**!

During the exam it is prohibited to:

1. Use any books or notes.
2. Use any electronic aid, including calculators.
3. Exchange information with any person other than the exam proctor.
4. Leave the exam room before you turn in your exam.

It is strongly suggested that you use all the time available. If you finish early, double check your work. By signing below you acknowledge that you have read and understood all of the instructions above.

Good luck!

NJIT Academic Honor Code Agreement

On my honor, I pledge that I have not violated
the provisions of the NJIT Academic Honor Code.

Name: _____ SID: _____

Signature: _____ Section: _____

Problem 1.

Convert the following infix expressions to prefix form by using the method discussed in class.

a. $a - b + c$ _____

b. $a/(b * c)$ _____

c. $(a + b) * c$ _____

d. $a - (b + c)$ _____

e. $a - (b/c * d)$ _____

f. $a/b/c - (d + e) * f$ _____

g. $a * (b/c/d) + e$ _____

h. $a - (b + c * d)/e$ _____

Problem 2.

Consider the following recursive definition for the Ackermann function:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

Write a function called `ack` that implements the above and then evaluate it for `ack(1,2)`.

Problem 3.

Recall the array-based implementation of the `Queue` class. In the solution presented the problem of rightward drift is solved by using a circular array. However, this solution did not allow the size of the queue to expand beyond its initial size.

```
public class Queue<E> {
    ...
    private void enqueue(E data) { ...  }
    private Object[] queue = new Object[10];
    private int head;
    private int tail;
}
```

You must implement the `enqueue` method so that, if it is necessary, the `queue` array can be expanded to accommodate more data items.

Problem 4.

Consider the following alphabet and language:

$$\Sigma = \{0, 1\}$$
$$L = \{w : w \in \Sigma^* \text{ and } w = w'\}$$

where w is a string and w' is the reverse of that string. Implement a recursive method (not a whole class) that recognizes strings in this language.

Problem 5.

Consider a slight variation on the queue. In this version new items can be added to and removed from either end. This data structure is commonly called a doubly-ended queue, or deque. Implement one method of each type ("insert" and "remove") given in the `Deque` class. You may assume the `DequeException` class has been defined for you.

```
public class Deque<E> {
    private class Node<T> {
        private Node(T data) {
            this.data = data;
        }
        private T data;
        private Node<T> next;
        private Node<T> prev;
    }
    public void insertAtHead(E data) {...}
    public void insertAtTail(E data) {...}
    public E removeFromHead() throws DequeException {...}
    public E removeFromTail() throws DequeException {...}
    private Node<E> head;
    private Node<E> tail;
}
```