DUKE UNIVERSITY
Department of Computer Science

Test 1: CompSci 100e

Name (print): _____

Community Standard acknowledgment (signature): _____

	value	grade
Problem 1	20 pts.	
Problem 2	13 pts.	
Problem 3	9 pts.	
Problem 4	8 pts.	
Problem 5	18 pts.	
TOTAL:	68 pts.	

This test has 10 pages, be sure your test has them all. Do NOT spend too much time on one question — remember that this class lasts only 75 minutes and there are 68 points on the exam. That means you should spend no more than *1 minute per point*.

You may consult your four (4) sheets and no other resources. You may not use any computers, calculators, or cell phones. You may refer to any program text supplied in lectures or assignments.

Don't panic. Just read all the questions carefully to begin with, and first try to answer those parts about which you feel most confident. Do not be alarmed if some of the answers are obvious.

If you think there is a syntax error or an ambiguity in a problem's specification, then please ask.

In writing code you do not need to worry about specifying the proper `import` statements. Assume that all libraries and packages we've discussed are imported in any code you write.

PROBLEM 1 : (*Evaluation (20 points)*)

A. Consider the following lines of code:

```
Integer i = null;
Integer j = null;
Integer k = null;
i= new Integer(72);
j=i;
k= new Integer(72);
```

What is the value of the following expressions:

I. `i == j`

II. `i == k`

III. `i.equals(k)`

IV. `72 == j.intValue()`

B. Evaluate each of the following expressions:

I. `(((13 > 0) || (4 < 3)) && (21 % 7 == 0))`

II. `7 * 8 % 10 / (-2 * -2.0)`

C. Given the following recursive method `mystery`:

```
int mystery(int n)
{
    if (n < 0)
     return -mystery(-n);
 else if (n < 10)
     return n;
 else
     return mystery(n/10 + n % 10);
}
```

What are would following calls evaluate to?

I. `mystery(7)`

II. `mystery(-512)`

D. Draw the list (i.e. that is a box-and-pointer diagram) resulting from the following code.

```
Node l = new Node("1", new Node("2", new Node("3", new Node("4", null))));
Node m = l.next.next;
Node n = new Node("5", m);
```

Your answer should clearly note what `l`, `m`, and `n` point to.

PROBLEM 2 : (*Big-Oh (13 points)*)

A. Given the following code,

```
public int clunk(int n)
{
  int sum = 0;
  for(int i=0; i < n; i++)
  {
    sum = sum + 10;
    for(int j=0; j <= i; j++) {
      sum = sum + 1;
    }
  }
  return sum;
}
```

I. What is the value of `clunk(3)`?

II. What is the running time of the `clunk`, in terms of n, using O-notation?

B. For each of the following, either indicate that it is true and *briefly* tell why or give a counterexample.

I. A $O(n)$ algorithm is always preferable to a $O(n^2)$ algorithm.

II. $n \log n + 4n^2 \in O(n^2)$

III. $3^n \in O(2^n)$

PROBLEM 3 : (*Sets (9 points)*)

A. What is $\log_2 1000$ to the nearest integer?

B. A set of values is maintained as a linked list, sorted in increasing order. What are the tightest asymptotic bounds you can give for the best and worst-case times for performing N insertions into this set, starting from an empty set? We want bounds for the worst-case time *and* bounds for the best-case time.

C. A set of values is maintained as an array, sorted in increasing order, plus an integer variable containing the current number of items in the set (which may differ from the size of the array). What are the tightest asymptotic bounds you can give for the best and worst-case times for performing N insertions into this set, starting from an empty set? Justify your answer. (Assume an ideal computer where N can be arbitrarily large and is not known in advance, and at any given time, the array has some finite size. Assume also that comparisons on the objects in the array require constant time.)

PROBLEM 4 : (*Matrix*)

Write the method `transpose` that returns the transposition (rows and columns swapped) of a rectangular two-dimensional array.

That is, if `mat` is $\begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{pmatrix}$ then `transpose` should return $\begin{pmatrix} 0 & 4 & 8 \\ 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \end{pmatrix}$

```
public static int[][] transpose(int[][] mat)
{
```

PROBLEM 5 : (*Strands*)

We would like to keep track of long strands of characters, (e.g. DNA), using a class, so we create the `IStrand` interface.

```
interface IStrand
{
    public void prepend(String s);
    public int length();

}
```

Below are two classes, `StringStrand` and `LinkStrand` that implement this interface.

```
class StringStrand implements IStrand
{
    private String myString;
    public StringStrand(){
     myString = "";
    }

    public void prepend(String s){
      myString = s + myString;
    }

    public int length(){
    return myString.length();
    }
}

class LinkStrand implements IStrand
{

    private Node myFront;
    private int myCount;

    public LinkStrand(){
  myFront = null;
  myCount = 0;
    }

    public void prepend(String s){
  myFront = new Node(s,myFront);
  myCount += s.length();
    }

    public int length(){
  return myCount;
    }

}
```

A. Which class do you think would be faster when performing n `prepend` operations? Briefly explain why.

B. Instead of just prepending to a Strand, we would like to splice in a String and insert it anywhere. That is, we would like to have `s.insert(k,str)` to add str at the kth position. Prepending coule now also be performed as `s.insert(0,str)`. We would like this behavior in both `LinkStrand` and `StringStrand`. What do we need to modify in `IStrand`?

C. Write the `insert` method for `StringStrand` below.

D. Write the `insert` method for `LinkStrand` below.

E. Which `insert` method will be faster for large strands? Briefly explain why.

Throughout this test, assume that the following classes and methods are available. These classes are taken directly from the material used in class. There should be no methods you have never seen before here.

List Node

```
public class Node
{
    String info;
    Node next;
    Node(String s, Node link) {
        info = s;
        next = link;
    }
}
```

String

```
public class String {
    /* Compares this string to the specified object.
     * The result is true if and only if the argument
     * is not null and is a String object that
     * represents the same sequence of characters */
    public boolean equals(Object anObject)

    /* Returns the index within this string of the
     *  first occurrence of the specified substring.
     * -1 if it does not exist */
    public int indexOf(String str)

    /* Returns the length of this string. */
    int length()

    /* Returns a new string that is a substring of this
       string. Begins at the specified beginIndex and
       extends to the character at index endIndex - 1 */
    String substring(int beginIndex, int endIndex)
 }
```

TreeSet/HashSet

```
public class TreeSet {
    // Constructs a new, empty set
    public TreeSet()

    // Returns an iterator over the elements in
    // this set.  The elements are visited in
    // ascending order.
    public Iterator iterator()

    // Returns the number of elements in this set.
    public int size()

    // Returns true if this set contains o
    public boolean contains(Object o)

    // Adds the specified element to this set
    // if it is not already present.
    public boolean add(Object o)
}
```

Integer

```
public class Integer
{
    /* Constructs a newly allocated Integer object that
       represents the specified int value. */
    public Integer(int value)
    /* Returns the value of this Integer as an int. */
    public int intValue()
}
```

ArrayList

```
public class ArrayList {
    // Constructs an empty list
    public ArrayList()
    // Returns the number of elements in this list.
    public int size()
    /* Searches for the first occurence of the given
       argument, returns  -1 if not found */
    public int indexOf(Object elem)
    // Returns the element at the specified position
    // in this list.
    public Object get(int index)
    /* Replaces the element at the specified position
     * in this list with the specified element. */
    public Object set(int index, Object element)
    // Appends the specified element to the end of
    // this list.
    public boolean add(Object o)
    // Inserts the specified element at the specified position
    public void add(int index, Object o)
    // Returns Iterator over elements in list
    public Iterator iterator()
}
```