**CSCI 135 Exam #2** **Name: _____**
**Fundamentals of Computer Science I**
**Fall 2012**

This exam consists of 6 problems on the following 6 pages.

You may use your two-sided hand-written 8 ½ x 11 note sheet during the exam. **You may use a calculator**, but no other computing or communication devices of any kind are permitted.

If you have a question, raise your hand and I will stop by. Since partial credit is possible, **please write legibly and show your work**.

| Problem | Points | Score |
|---|---|---|
| 1 | 7 | |
| 2 | 12 | |
| 3 | 10 | |
| 4 | 7 | |
| 5 | 10 | |
| 6 | 9 | |
| Total | 55 | |

**1. Loops and conditionals** (7 points). Consider the following program:

```
public class Mystery
{
    public static void main(String [] args)
    {
        final int N = Integer.parseInt(args[0]);

        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                if ((i == 0) || (i == N-1) || (j == 0) || (j == N-1))
                    System.out.print("*");
                else
                    System.out.print("-");
            }
            System.out.println();
        }
    }
}
```

Give the output of the following commands:

```
% java Mystery 1

*


% java Mystery 2


**
**


% java Mystery 5

*****
*---*
*---*
*---*
*****
```

**2. Multiple choice** (12 points, 2 points each). Circle the ***best*** single answer.

I. All of the following statements result in a compile error ***EXCEPT***:

```
a) ArrayList<double>  nums = new ArrayList<double>();
b) ArrayList<Integer> nums = new ArrayList<Integer>;
c) ArrayList<Integer> nums = new ArrayList<Integer>();
d) ArrayList          nums = new ArrayList(int);
```

II. Which of the following most accurately describes the job of Java's garbage collector:

a) Allocates memory to newly instantiated objects
b) Removes non-ASCII characters from String objects
c) Protects the programmer from null pointer exceptions
d) Frees up memory of instantiated objects that are no longer referenced anywhere.

III. You have a for-loop that appends to a `String` object during every iteration of the loop. For long strings, the loop takes a long time to run. What is the best solution?

a) Increase the heap size available to the Java Virtual Machine (JVM).
b) Switch to the `StringBuilder` data type, it is not immutable and supports efficient appending.
c) Change your for-loop to a do-while loop.
d) Change the for-loop to go backwards, appending to the front of the `String`.

IV. You want to sort some data and then search it quickly. Assuming the implementation of all algorithms have a constant term of 1, which pair of algorithms would be the fastest?

a) merge sort, binary search
b) insertion sort, binary search
c) merge sort, linear search
d) insertion sort, linear search

V. Queues and stacks are two fundamental abstract data types (ADTs) in computer science. Which of the following statements is ***FALSE***:

a) Queues implement a first-in first-out (FIFO) removal policy.
b) Stacks implement a last-in first-out (LIFO) removal policy.
c) Both queues and stacks can be implemented using arrays.
d) Stacks can be implemented using an array, but queues cannot.

VI. Which of the following statements is ***TRUE***:

a) A Java class must always have at least one constructor declared.
b) A recursive method can always be implemented as a non-recursive method.
c) A double variable will promote to an int when necessary, but not vice versa.
d) All object variables of the same class share the same values for their instance variables.

3. **Conditionals** (10 points). Complete the Java program `Streak.java` that reads in a sequence of ***positive*** integers from standard input and prints out the length of the longest streak (consecutive values in the sequence that are the same value). Assume you have access to the `StdIn` class.

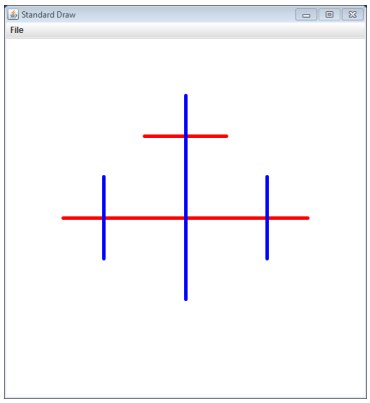
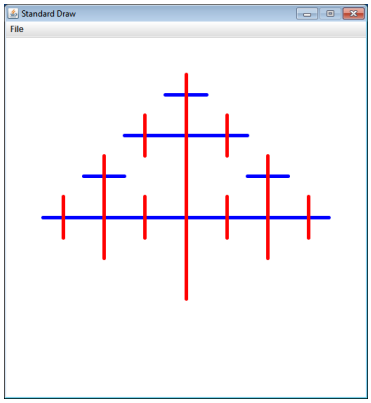
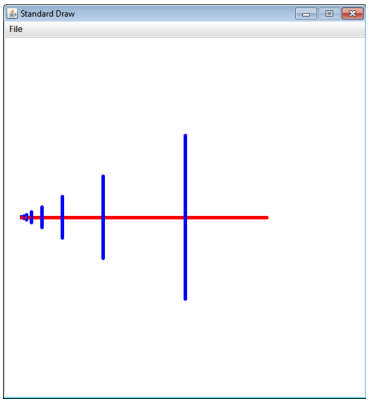
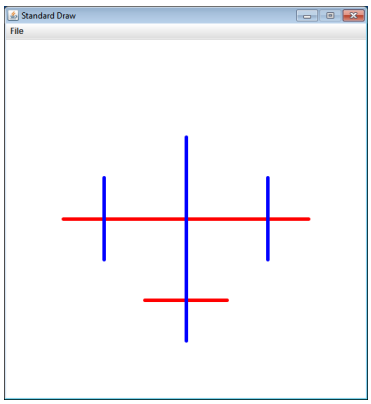
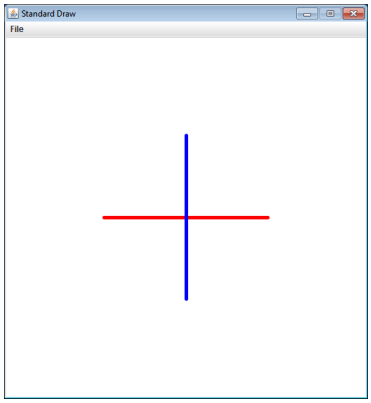
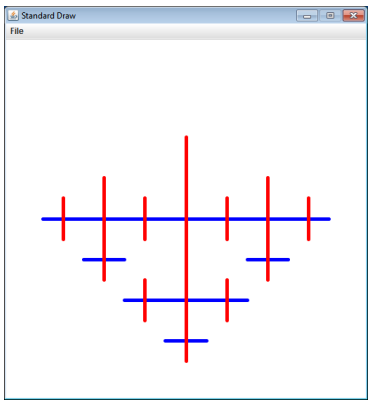
```
Example run:

% more input.txt
1 5 5 5 7 2 2 1 5 2 4

% java Streak < input.txt
3
```

```
public class Streak
{
   public static void main(String [] args)
   {
      int longest = 0;  // Length of the longest streak thus far
      int current = 0;  // Length of the current streak
      int last    = -1; // Previous value, -1 = flag value for start

      while (!StdIn.isEmpty())
      {
          int num = StdIn.readInt();



          if (num == last)
              current++;
          else
              current = 1;
          if (current > longest)
              longest = current;

















          last = num; // Save off this value for use in the next loop
      }

      System.out.println(longest);
   }
```

**4. Recursive graphics** (7 points). Consider the following recursive drawing program:

```
public class RecursiveGraphics
{
    public static void draw(int n, double x, double y, double size)
    {
        if (n == 0) return;
        StdDraw.setPenColor(StdDraw.RED);   StdDraw.line(x-size, y     , x+size, y     );
        StdDraw.setPenColor(StdDraw.BLUE);  StdDraw.line(x     , y-size, x     , y+size);
        draw(n-1, x-size, y     , size/2.0);
        draw(n-1, x+size, y     , size/2.0);
        draw(n-1, x     , y+size, size/2.0);
    }
    public static void main(String [] args)
    {
        StdDraw.setPenRadius(0.01);
        draw(Integer.parseInt(args[0]), 0.5, 0.5, 0.25);
    }
}
```

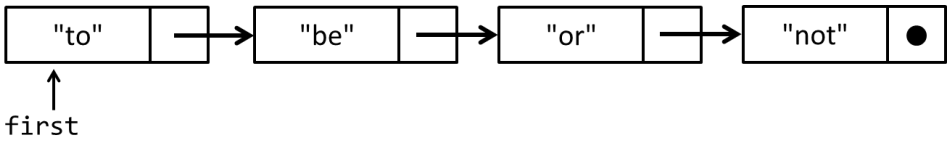
Below are a set of possible results, only three were generated by the above program. Label the 3 results that were generated by the above program, labeling the result with the `args[0]` value that *could* generate it.

<table>
<tr>
<td>  <br>2</td>
<td>  </td>
<td>  <br>Any negative int, large positive int</td>
</tr>
<tr>
<td>  </td>
<td>  <br>1</td>
<td>  </td>
</tr>
</table>

**5. Linked structures** (10 points). You have a linked list that holds strings. It uses the following inner class Node:

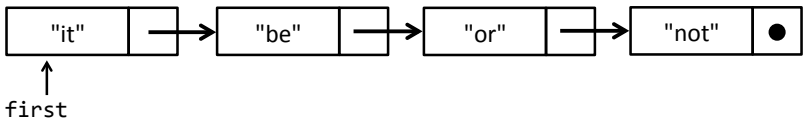
```
private class Node
{
    String item;
    Node next;
    Node(String s, Node n) { item = s; next = n; }
};
```

Currently the linked list has the following data and structure:

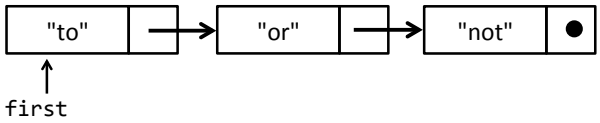



Draw the linked list resulting from running each of the following code segments. Be sure to ***show where the `first` variable*** is pointing. ***Each part is independent*** of the other parts (assume each part starts with the linked list shown in the above diagram).

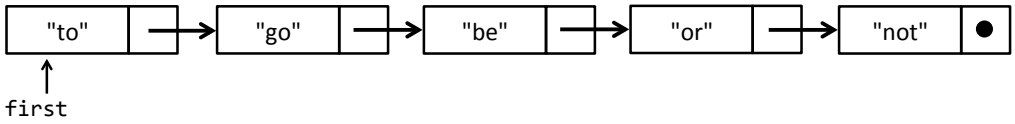
```
first = new Node("it", first.next);
```



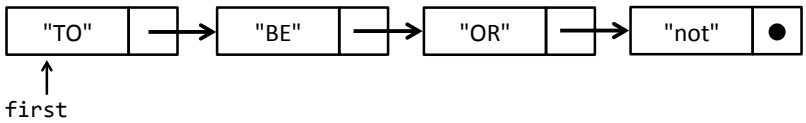

```
first.next = first.next.next;
```



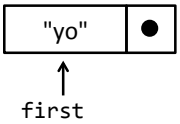

```
first.next = new Node("go", first.next);
```




```
for (Node c = first; c != null; c = c.next)
{
    if (c.item.length() <= 2)
        c.item = c.item.toUpperCase();
}
```




```
first = new Node("yo", null);
```



**6. Performance** (9 points). The following table gives running times measured for a program using an input size of N, for various values of N.

| N | time (seconds) |
|---|---|
| 50 | 0.63 |
| 100 | 4.33 |
| 200 | 33.69 |
| 400 | 263.82 |

a) Which of the following best describes the order of growth of the running time of this program? ***Circle one*** of the following:

I. O(1), constant
II. O(log N), logarithmic
III. O(N log N), linearithmic
IV. O(N), linear
V. $O(N^2)$, quadratic
VI. $O(N^3)$, cubic $263.82 / 33.69 \approx 7.83081 \approx 8 = 2^3$
VII. $O(2^N)$, exponential

b) Give the equation showing the running time of the program in seconds as a function of the input size N (you need to solve for the leading constant).

$T = a N^3$
$263.82 = a (400)^3$
$a = 4.12219 \times 10^{-6}$
$T = 4.12219 \times 10^{-6} N^3$

c) Estimate the program's running time in seconds for an input size of N = 1000.

$T = 4.12219 \times 10^{-6} (1000)^3$
$T = 4122.1875$ seconds