COS 126	General Computer Science	Fall 2004

Exam 2

This test has 10 questions worth a total of 50 points. You have 120 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write out and sign the Honor Code pledge before turning in the test.**

"I pledge my honor that I have not violated the Honor Code during this examination."

Signature

Problem	Score
1	
2	
3	
4	
5	
Sub 1	

Problem	Score
6	
7	
8	
9	
10	
Sub 2	

Total	

Name:

NetID:

Precept:

1	MF 10:00	Donna
2	MF 11:00	Eddie
3	MF 1:30	Chris B.
4	MF 2:30	Kevin
5	M 7:30 F 2:30	Chris D.

1. **Boolean circuits. (5 points)**

A structure frequently used in computers to transport data between different devices is the *bus*. At any time only one device is allowed to transmit data over the bus, and a special circuit called the *bus arbiter* decides which component is granted permission to use it.

(a) Give the truth table of a bus arbiter which arbitrates bus access for 3 devices with IDs 0, 1 and 2. If input REQ_i is true, the device with ID i is requesting permission to use the bus. The arbiter uses output GRA_i to signal back if permission has been granted to device i (GRA_i is true in that case). If two or more devices are requesting access permission at the same time, the bus arbiter grants permission to the device with the lowest ID.

REQ_0	REQ_1	REQ_2	GRA_0	GRA_1	GRA_2
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

(b) Draw a circuit for the bus arbiter using only AND and NOT gates.

2. **Analysis of algorithms. (4 points)**

   Estimate each of the following quantities. Circle the *best* answer.

   (a) Number of *bits* in the binary representation of 78^{123}.

   14 44 124 541 775

   (b) Maximum `N` for which the following code fragment finishes in under one minute on Kevin's laptop.

```
public static long f(long N) {
    if (N <= 2) return N;
    else return f(N-1) + 3*f(N-2) + f(N-3);
 }
```

   11 33 95 17322 $91,131,077$

   (c) Value of `count` upon termination of the nested loop when N = 1,000.

```
int count = 0;
for (int i = 0; i < N; i++)
    for (int j = i + 1; j < N; j++)
        count++;
```

   $1,000$ $9,996$ $499,500$ $1,000,000$ $1,000,000,000$

   (d) Number of string comparisons to quicksort 5,589,825 words from the works of Charles Dickens.

   2,364 5,589,825 137,775,671 13,215,917,463 31,246,143,530,625

3. **Data types. (7 points)**

   Create an abstract data type for charged particles. Each charged particle i has a position (x_i, y_i) and a charge q_i. The *electric potential* at (x, y) due to particle i is given by $V_i = kq_i \,/\, r$ where r is the distance between (x, y) and particle i, and $k = 8.99 \times 10^9$ is the *electrostatic constant*. (All values are measured in standard MKS units.) Your data type should have a constructor and a method `potential(x, y)` that returns the electric potential at (x, y) due to particle i. Use a helper method `distanceTo(x, y)`.

```
public class ChargedParticle {




   // create a charged particle at (x, y) with charge q
   public ChargedParticle(double x, double y, double q) {




   }

   // return the Euclidean distance from the invoking charge to (x, y)
   public double distanceTo(double x, double y) {




   }

   // return the electrical potential at (x, y) from invoking charge
   public double potential(double x, double y) {




   }
}
```

4. **Strings and regular expressions. (6 points)**

   (a) What does the following code fragment print out? Circle your answer. Recall that `s.replaceAll(re, b)` replaces all occurrences of the regular expression `re` in `s` with `b` and returns the result.

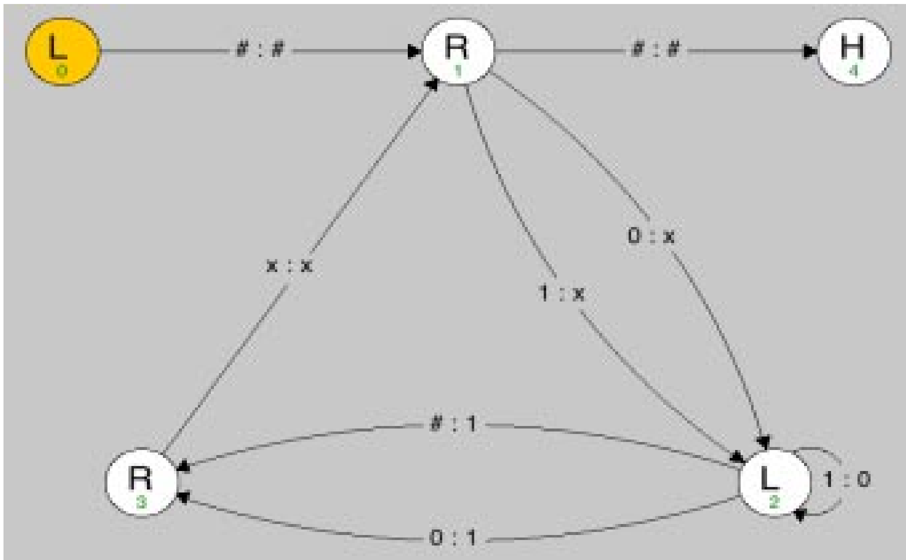
```
String s = "CAAGAATTGA";
s = s.replaceAll("A", "T");
s = s.replaceAll("C", "G");
s = s.replaceAll("G", "C");
s = s.replaceAll("T", "A");
System.out.println(s);
```

   (b) Write a Java regular expressions that identifies whether a line in a file is comprised of a sequence of positive integers, separated by commas with no intervening whitespace. Circle your answer.

```
Valid                                 Invalid
--------------------------------------------------------
56,743434343433433343434 7,44,33      78,abcd,44,33,22
67,1,4,3,2                            67,0,4,3,2
                                      78,-55,33,22
7                                     seven
1,2,3,4                               1, 2, 3, 4
8,7,6,5,4,30,9                        ,,,,,
```

5. **Turing machines. (5 points)**

The Turing machine below uses the following four character alphabet: `0 1 # x`.



(a) If started with the input tape and cursor shown below, what will be left on the tape when the machine halts?

```
# # # # # # # # # # # 1 1 # # # # # # # # # # # # #
                      ^
```

(b) If started with the input tape and cursor shown below, what will be left on the tape when the machine halts?

```
# # # # # # # # # # # 1 0 1 0 # # # # # # # # # # #
                      ^
```

(c) Describe in words what this Turing machine does, when started with an N-bit binary number x.

(d) About how many steps will it take for the machine to halt when started with an N-bit input. Circle the best answer.

N $\quad$ $N \log N$ $\quad$ N^2 $\quad$ N^3 $\quad$ 2^N

This page is for scratch work.

6. **Cryptography. (4 points)**

For each problem on the left, put the letter of the *best* matching *guarantee* on the right. You may use an answer more than once.

___ Determine Bob's private RSA key (d, N), given Bob's public RSA key (e, N), an RSA encrypted message from Alice, and the original unencrypted message.

___ Determine Bob's private RSA key (d, N), given Bob's public key (e, N) and a factorization of $N = p \times q$.

___ Determine Alice's original message, given Bob's public RSA key (e, N) and Alice's RSA encrypted message to Bob.

___ Decrypt a message sent with a one-time pad without knowing the one-time pad key.

A. Solvable in a polynomial time.

B. Solvable in polynomial time if factoring can be solved in polynomial time.

C. Solvable in polynomial time if P = NP.

D. Solvable in exponential time.

E. Unsolvable: there is no algorithm to solve this problem.

7. **Intractability. (4 points)**

Suppose that problems X and Y are each NP-complete. Circle each of the following statements that is true.

I. If X is solvable in polynomial time then so is Y.

II. If X is not solvable in polynomial time then neither is Y.

III. If X is solvable in polynomial time then $P = NP$.

IV. If X is not solvable in polynomial time then $P \neq NP$.

8. **Symbol tables. (4 points)**

   Suppose that you have a symbol table `st` of key-value pairs, where the keys are misspelled words and the values are the corrected versions. Here is a partial list of key-value pairs: teh-the, mispell-misspell, morgage-mortgage. Write a Java fragment that reads in words from standard input and prints them back out, correcting any misspelled words. Recall that a `SymbolTable` has the following interface:

   - `put(key, value)`: insert the key-value pair
   - `get(key)`: if the key was previously inserted, return the corresponding value; otherwise return `null`

9. **Linked structures. (7 points)**

Complete the implementation for a randomized queue data type.

```
public class RandomizedQueue {
   private Node first;
   private class Node {
       String value;
       Node next;
   }

   // create an empty randomized queue
   public RandomizedQueue() { first = null; }

   // insert string s into the randomized queue
   public void insert(String s) {
       Node x = new Node();
       x.value =
       x.next  =
       first   =
   }

   // return the number of elements in the randomized queue
   public int size() {






   }

   // return and delete a *random* element from the queue
   public String delete() {









   }
```

10. **References. (4 points)**

This problem refers to the `RandomizedQueue` data type in the previous question, although this question can be answered independently. What does the following program do?

```
public static void main(String[] args) {
   RandomizedQueue a = new RandomizedQueue();
   RandomizedQueue b = new RandomizedQueue();
   RandomizedQueue c = new RandomizedQueue();

   a.insert("last");
   a.insert("question");
   b.insert("last");
   b.insert("question");
   c.insert("hurray!");
   c.insert("hurray!");
   if (a == b) System.out.println("a == b");
   c = b;
   b = a;
   a = c;
   b = a;
   c = b;
   if (a == c) System.out.println("a == c");
   while (a.size() > 0) {
       b.insert(a.delete());
   }
   System.out.println("goodbye");
}
```