# Computer Science II

# Summer 2014

# Final Exam

# Instructor: Arup Guha

# Date: 7/31/2014

**Name: _____**

1) (12 pts) Coordinate Geometry

(a) (9 pts) Let line segment A be between (2, 5) and (-4, 17). Let line segment B be between (-3, 9) and (4, 16). Using the vector/parametric equations corresponding to the two line segments, determine the parameter of intersection on both line segments and find the point of intersection. (Note: All of your grade is based on using the vector/parametric equation of the lines.)

(b) (3 pts) In the method shown below, used in the solution to the Birdman problem, explain why there is no danger of a divide by 0 error in the else clause.

```
public double getLambda(pt dest) {

      if (dir.x != 0)
            return 1.0*(dest.x - start.x)/dir.x;

      else
            return 1.0*(dest.y - start.y)/dir.y;
}
```

2) (10 pts) String Matching

(a) (5 pts) Consider using the Boyer-Moore string matching algorithm with the pattern "abracadabra". Fill in the last array for the letters 'a', 'b', 'c', 'd', and 'r'. Please use zero-based indexing. (Thus, all blanks should be filled in with an integer in between 0 and 10, inclusive.)

```
last['a'] = ____

last['b]] = ____

last['c'] = ____

last['d'] = ____

last['r'] = ____
```

(b) (5 pts) Consider using the Knuth-Morris-Pratt string matching algorithm with the pattern "ababbababb". Fill in the failure array the algorithm creates from the pattern string below:

| **index** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|---|
| **letter** | a | b | a | b | b | a | b | a | b | b |
| **fail** | | | | | | | | | | |

3) (10 pts) Dynamic Programming (Matrix Chain Multiplication)

Using the dynamic programming algorithm shown in class, determine the minimum number of multiplications to calculate the matrix product ABCD, where the dimensions of A, B, C and D are given below:

A: 2 x 5, B: 5 x 4, C: 4 x 1, D: 1 x 5

| | A | B | C | D |
|---|---|---|---|---|
| A | 0 | | | |
| B | X | 0 | | |
| C | X | X | 0 | |
| D | X | X | X | 0 |

4) (10 pts) Number Theory

(a) (5 pts) Imagine testing 817 for primality using the Miller-Rabin primality test. In a single test, you'll successively calculate a randomly chosen base a raised to various powers mod 817. What are each of those powers?

(b) (5 pts) In an RSA system, $p = 17$ and $q = 11$. Determine n and $\varphi(n)$.

5) (9 pts) Dynamic Programming – Longest Increasing Sequence

Assume that you've already written a method LCS, that calculates the length of the longest common subsequence between two sequences of integers. (Its prototype is given below.) Write a method that takes in 1 sequence of integers and calculates its longest increasing sequence. For ease of implementation, assume that the input only contains distinct integers. Note: You may use both the Arrays.copyOf and Arrays.sort methods. These are listed below. (Note: The code is pretty short.)

```
// Returns an array storing the first newLength elements of original.
int[] copyOf(int[] original, int newLength);

// Sorts the specified array into ascending numeric order.
void sort(int[] a);


public static int lis(int[] seq) {









}

public static int lcs(int[] x, int[] y) {

    int[][] table = new int[x.length+1][y.length+1];

    for (int i = 1; i<=x.length; i++) {
        for (int j = 1; j<=y.length; j++) {
            if (x[i-1] == y[j-1])
                table[i][j] = 1+table[i-1][j-1];
            else
                table[i][j] = Math.max(table[i][j-1], table[i-1][j]);
        }
    }
    return table[x.length][y.length];
}
```

6) (12 pts) Dynamic Programming – Zero/One Knapsack Problem

Find the maximum valued knapsack of size 12 or less choosing from the following items (only 1 copy of each item is available). To get credit, please use the algorithm shown in class. Here are the items from which you are choosing:

| Item | Weight | Value |
|---|---|---|
| Apple | 3 | 4 |
| Banana | 4 | 6 |
| Cantaloupe | 7 | 13 |
| Durian | 5 | 9 |
| Emblic | 2 | 5 |
| Fig | 1 | 3 |

Show the algorithm by filling in the following table:

| Item | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Apple | | | | | | | | | | | | |
| Banana | | | | | | | | | | | | |
| Cantaloupe | | | | | | | | | | | | |
| Durian | | | | | | | | | | | | |
| Emblic | | | | | | | | | | | | |
| Fig | | | | | | | | | | | | |

The correct answer to the query should be in the bottom right square of the table.

7) (10 pts) Individual Programming Assignment Details

(a) (5 pts) Several students missed one case on program 4 involving Dijkstra's Algorithm. Theoretically, their code was sound, but one of their output values was off by .01. (Remember that it was requested that the output value be rounded to the nearest hundredth using the usual rules for rounding.) Give an example of an actual shortest distance that is likely to have caused this error. Note: Obviously, I don't expect you to remember this case. The idea is that you should be able to create a separate case by hand that is likely to recreate this same sort of error. For this problem, all you have to do is list the corresponding shortest distance for such a case and explain WHY this is can possibly cause an error if the programmer isn't careful.

(b) (5 pts) In program #6, you were asked to write a dynamic programming solution to a "Tolls" problem. Ultimately, when breaking the problem down recursively, there were only two important cases to consider at each toll booth. Explain, in words, what are those two cases are.

8) (10 pts) Graphs – Topological Sort

Alice has to complete items 1 through 10. The following ordered pairs show the dependency between the items she must complete. Namely, for each ordered pair (a, b) shown below, she must complete item a before item b.

(2, 7), (8, 3), (9, 2), (4, 5), (4, 1), (4, 7) and (6, 10).

(a) (5 pts) Show the ordering of the items produced by the algorithm shown in class that utilizes a depth first search (DFS). Iterate through the nodes in numeric order when running each DFS. Furthermore, within any DFS, if there is a choice to travel to multiple nodes, go to the lower numbered node first. Finally, remember to fill in the slots in the ordering in backwards order, as this algorithm requires.

_____ , _____ , _____ , _____ , _____ , _____ , _____ , _____ , _____ , _____

(b) (5 pts) Show the ordering of the items produced by the algorithm shown in class that builds the list from the front and always adds "safe" nodes iteratively. When choosing between multiple possible "safe" nodes, always add the lowest numbered one first.

_____ , _____ , _____ , _____ , _____ , _____ , _____ , _____ , _____ , _____

9) (15 pts) Algorithm Design – Dynamic Programming

Describe a dynamic programming algorithm (in words) to solve the following problem:

Soccer players are ordered 0, 1, 2, …, n – 1, where n is a positive integers less than or equal to 100. Each player i ($0 \leq i \leq n-2$) can pass to any player j such that $j > i$. The probability the pass will succeed is prob[i][j]. (Assume that this information is given in the input.) As with all probabilities, note that $0 \leq \text{prob}[i][j] \leq 1$, for all $0 \leq i < j \leq n - 1$.

We are allowed to choose any sequences of passes in order to move the ball from player 0 to player n – 1. (This sequence will necessarily be some subsequence of 0, 1, 2, …, n – 1, since all the players can only pass the ball in "one direction.") If all players act optimally, design an algorithm that calculates the probability that player n – 1 successfully receives the ball.

10) (2 pts) For Fun

With what liquid are water balloons typically filled? _____

**Scratch Page – Please clearly mark any work on this page you would like graded.**