Computer Science II Exam #2, July 3, 2014

Last Name: _____, First Name: _____

1) (15 pts) Huffman Coding

Consider a file that contains the following eight distinct characters (each encoded using 3 bits) with the following frequencies:

Character	Frequency	Huffman Code
A	3000	
B	8000	
C	14000	
D	10000	
E	25000	
F	9000	
G	18000	
H	13000	

a) (5 pts) Draw a Huffman Coding Tree for this file below.

b) (5 pts) In the chart above, fill in the corresponding Huffman codes for each character in the file based on the tree you drew in part a.

c) (5 pts) How many bits would be saved using the Huffman coding on this file, assuming that the code itself took 100 bits to store?

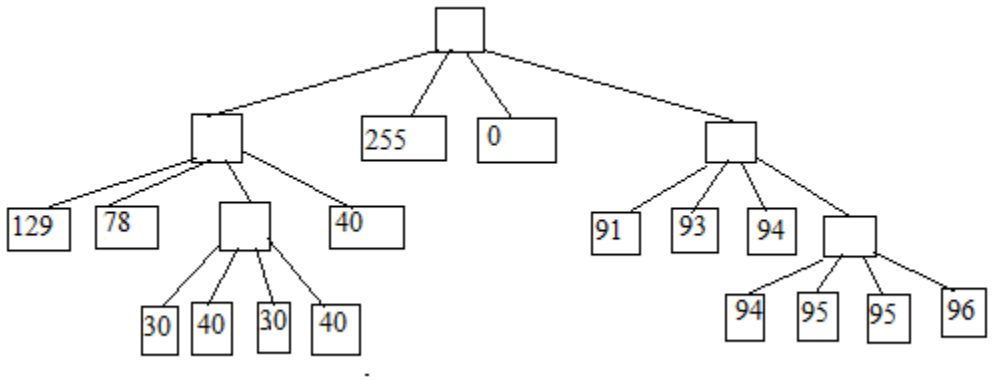
2) (18 pts) Divide and Conquer Code – QuadTrees

A quadtree is a data structure to store an image. For simplicity, we assume the dimensions of the image to be 2^k x 2^k, for some non-negative integer k. If each pixel in the whole image has the same value, we store this entire picture as a single node with that value. On the other hand, if this image has at least two distinct values in its pixels, we store no color value in the root node of the tree, but instead store four pointers to the upper left, upper right, lower left and lower right quadrants of the image. These pointers are simply pointing to quadtrees that represent that portion of the image. This alternate storage scheme for a picture can be seen well visually. Imagine we have the following 8 x 8 grayscale image:

129	129	78	78	255	255	255	255
129	129	78	78	255	255	255	255
30	40	40	40	255	255	255	255
40	30	40	40	255	255	255	255
0	0	0	0	91	91	93	93
0	0	0	0	91	91	93	93
0	0	0	0	94	94	94	95
0	0	0	0	94	94	95	96

Note: A grayscale pixel stores an intensity of shade as in integer, ranging from 0 (white) to 255 (black).

We would store this image as a quadtree as follows:



One useful computation for a grayscale image is the average intensity of its pixels. (This might correspond to how much ink would be used to print it, for example.) Complete the method intensity in the quadtree class on the next page so that it returns a double, corresponding to the average value in the pixels of the designated quadtree image.

```
class quadtree {

    final public static int INTERNAL = -1;
    private int shade;
    private quadtree[] children;

    // Assume pic = 2^k by 2^k
    public quadtree(int[][] pic, int x, int y, int n) {

        if (same(pic, x, y, n)) {
            shade = pic[x][y];
            children = null;
        }

        else {
            shade = INTERNAL;
            children = new quadtree[4];
            children[0] = new quadtree(pic,x,y,n/2);
            children[1] = new quadtree(pic,x,y+n/2,n/2);
            children[2] = new quadtree(pic,x+n/2,y,n/2);
            children[3] = new quadtree(pic,x+n/2,y+n/2,n/2);
        }
    }

    // Requires that x >= 0, y >= 0, x+n <= pic.length, y+n <= pic.length
    public static boolean same(int[][] pic, int x, int y, int n) {
        int val = pic[x][y];
        for (int i=x; i<x+n; i++)
            for (int j=y; j<y+n; j++)
                if (pic[i][j] != val)
                    return false;
        return true;
    }

    public double intensity() {

        if ( _____________________________ )
            return shade;

        double ans = 0;
        for (int i=0; i<4; i++)

            ____________________________________________________ ;



        return ____________________;



    }
}
```

// Note: Since there is very little code here, each tiny bit is worth quite a few points with
// relatively few opportunities for partial credit.

3) (15 pts) Network Flow Trace

A flow network is described by the following directed edges and capacities:

AF 8	CA 10	EB 5	FD 20
AB 2	CB 20	ED 4	GB 7
BF 12	CG 15	EF 4	GE 7

a) (2 pts) What are the source and sink vertices, respectively?

Source = ____ Sink = ____

b) (4 pts) Draw the corresponding flow network. (Note: Spend a minute determining how to lay out your vertices to minimize the crossing of edges.)

c) (9 pts) Determine the maximum flow through this network as shown in class. Show each augmenting path you use and how much flow that path adds to the network. (Note: You may not use each line in this chart.)

Augmenting Path	Flow Added

Max Flow = _____

4) (12 pts) Dijkstra's Algorithm Trace

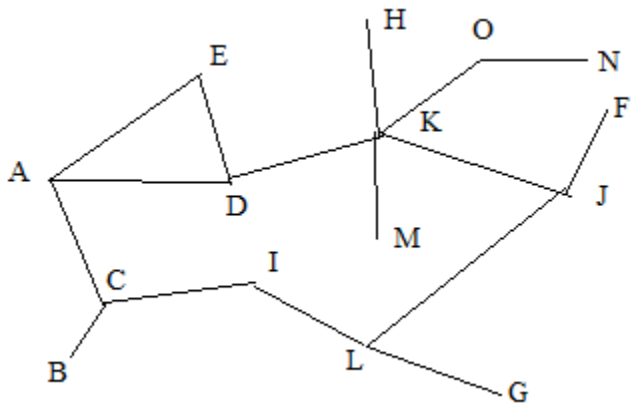
Trace through Dijkstra's algorithm on the graph described below using A as the source vertex. The last line of your chart should display the shortest distances from A to all of the vertices in the graph. In order to receive credit, you must properly fill in the chart, which shows incremental updates to the distance array. (Note: Each edge shown below is a directed edge.)

AB 12	BG 4	CG 4	EB 1	FE 1
AC 16	BC 1	DC 8	EG 8	FG 10
AD 5	CE 10	DF 4	FC 4	

Add to S	A	B	C	D	E	F	G
A	0						

5) (10 pts) Graphs – Breadth First Search

Show the order in which the vertices get visited during a breadth first search of the graph shown below, starting at vertex A. When enqueing multiple unvisited neighbors of a current node during the algorithm, always enqueue the neighbors in alphabetical order.



___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___

6) (15 pts) Graphs – Depth First Search

A x-y knight is a piece on an n x n board which must move exactly ±x units in the x-direction and ±y units in the y direction on a single jump. For example, a 3-4 knight can move from (10, 12) to the following locations: (7, 8), (7, 16), (13, 8), (13, 16). Given a starting spot for an x-y knight on an n x n board with valid indexes 0 through n – 1, inclusive, complete the code below to implement a depth first search to mark all the reachable squares on the board from the initial square.

```
import java.util.*;
public class knight {

    public static int x;
    public static int y;
    public static int n;
    public static boolean[][] reachable;

    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        n = stdin.nextInt(); // Board from (0,0) to (n-1,n-1).
        x = stdin.nextInt(); // Distance to move in x direction.
        y = stdin.nextInt(); // Distance to move in y direction.
        int startX = stdin.nextInt(); // Starting x position.
        int startY = stdin.nextInt(); // Starting y position.
        reachable = new boolean[n][n];
        dfs(startX, startY);

        for (int i=0; i<n; i++) {
            for (int j=0; j<n; j++)
                System.out.printf("%c ", reachable[i][j] ? 'T':'F');
            System.out.printf("\n");
        }
    }

    public static void dfs(int curX, int curY) {









    }

    public static boolean inbounds(int tryX, int tryY) {
        return tryX >= 0 && tryX < n && tryY >= 0 && tryY < n;
    }
}
```

7) (13 pts) Greedy Algorithm Design

A violinist puts fingers down on a string to play different notes. She can either play an open string (0), or put down her index finger (1), middle finger (2), ring finger (3) or pinkie (4). If she puts down multiple fingers, the note that plays is the one that corresponds to the highest number from 1 to 4. To make playing a sequence of notes easier, a violinist wants to minimize the sum of the number of times she moves her fingers from either on the string to off the string or off the string to on the string. Given a sequence of notes designated by a string of integers in the set {0,1,2,3,4}, representing the highest finger that must be placed on the string to produce the correct note, determine an algorithm to minimize the value described above. Use your algorithm to find the desired minimum value on the sequence shown below the example.

Consider the following example input sequence: 0, 0, 0, 0, 1, 1, 4, 3, 3, 2, 2, 1, 1, 0. Here we would put our 1^{st} finger (index finger) down for the fifth note, put down our 2^{nd}, 3^{rd} and 4^{th} fingers for the 7^{th} note, lift our 4^{th} finger for the eighth note, lift our 3^{rd} finger for the tenth note, lift our 2^{nd} finger for the 12^{th} note and lift our 1^{st} finger for the last note, for a total of 8 finger movements.

a) (10 pts) Describe your algorithm to solve this problem.

b) (3 pts) Trace through your algorithm on the following sequence, giving the minimum number finger movements to play the sequence:

1, 3, 3, 2, 0, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 3, 2, 3, 2, 3, 1, 1, 1, 1, 1, 2, 3, 1, 1

Minimum number of finger changes = _____

8) (2 pts) The U.S. lost its last World Cup game in spite of the heroic efforts of goalie Tim Howard, who had a record setting day with 16 saves against Belgium. Tim Howard's job is to prevent the opposing team from kicking the soccer ball into what location on the field?

_____

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