COP 3503 Recitation #4 Recursion Problems (Week of 6/10)
Due: 6/20/13 (Thursday) at 11:55 PM Webcourses2 time

Write complete Java program. Please include ample comments in your code. Using Big-O notation, indicate the time complexity in terms of the appropriate variables.

Problem: Graph conversion

For this recitation, you will write a program that will convert an adjacency matrix representation to an adjacency list representation, and vice versa.

The program will take file input, from the file 'conversions.txt'. The general form of the file will be as follows.

```
<conversion_type>
<graph representation>
<conversion type>
<graph representation>
…
…
<conversion type>
<graph representation>
0
```

<Conversion type> will be a single digit – either 1 or 2. 1 indicates a conversion from adjacency matrix to list, 2 indicates the conversion will be other way around. The 0 at the end is a dummy conversion type used to indicate end of input.

<Graph representation> will be an adjacency matrix (if conversion type is 1) or an adjacency list (if conversion type is 2).

An adjacency matrix is denoted as follows. The first line will give an integer *N*, indicating the number of vertices in the graph. This is followed by *N* lines, each containing *N* integers (either 1 or 0) – basically a straightforward representation of the adjacency matrix.

Adjacency lists are slightly more complicated to represent. The first line gives an integer *N*, indicating the number of vertices in the graph. This is followed by *N* lines. The *k*th line represents the neighbors of vertex *k* (*0 <= k <= n-1*, i.e., vertices use zero-based indexing) as a series of space-separated numbers. The first number on each line, *p*, gives the number of neighbors of that vertex. This is followed by *p* numbers: the vertices that are adjacent to the vertex. Note that these numbers are in no particular order, so don't assume that they will be sorted or any such thing.

For output, your program must perform the conversion (based on conversion type) and write the result to standard output. The format of the output will match the matrix and list

format described above. Note that you must include the first line that gives the number of vertices, just as in the input.

While the obvious way to do this is to explicitly build an adjacency list/matrix and convert, you don't necessarily need to go that far if you can code up something simpler. Whatever works.

What to turn in

Turn in the file Conversion.java to your TA via email. Please follow all the specifications and don't forget to put comments into the code.

Sample Input

```
1
4
0 0 1 0
0 0 1 1
1 1 0 1
0 1 1 0
2
6
2 1 4
3 2 0 5
2 1 3
2 2 4
3 5 0 3
2 4 1
0
```

Sample Output

```
4
1 2
2 2 3
3 0 1 3
2 2 1
6
0 1 0 0 1 0
1 0 1 0 0 1
0 1 0 1 0 0
0 0 1 0 1 0
1 0 0 1 0 1
0 1 0 0 1 0
```