CSC222 Data Structures & Algorithms II Spring 2014

Project 3: Strongly Connected Components

Your task: Implement the algorithm (discussed in class) for finding strongly connected components of a directed graph.

Input:

Your program must accept the name of a file on the command line. No default file name is necessary. If the file name is missing, or if too many file names are given, print an error message and exit.

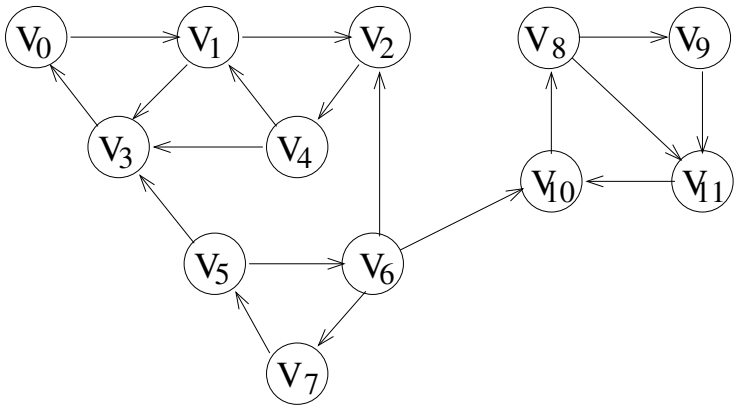
The first line of the input file contains the number of vertices n in a directed graph. Each subsequent line in the file contains a pair of numbers (u, v) which represents an edge from vertex u to vertex v (in that direction). Vertex numbering starts at zero, so $0 \leq u < n$ and $0 \leq v < n$ for each pair read. Your C++ program should continue reading up to end of file.

Sample test files (with correct output) can be found on our web site *http://menehune.opt.wfu.edu/csc222*.

Example Input:

```
12
0 1
1 2
1 3
2 4
3 0
4 1
4 3
5 3
5 6
6 2
6 7
6 10
7 5
8 9
8 11
9 11
10 8
11 10
```

The input example shown above corresponds to the directed graph illustrated below:



After successfully reading the input, your program should find the strongly connected components of the input directed graph. Number the components $1, 2, 3, ...$ Refer to the class handout entitled "Depth First Search of a Graph" for a pseudo-code description of how to compute strongly connected components. When you sort the nodes according to decreasing post-number, use an efficient variation on bucket sort. Your sort should be accomplished in time $\mathcal{O}(n)$, where n is the number of vertices in the graph.

Output:

For each strongly connected component found, output the vertices found in that component. For example:

```
Component 1:  8 9 10 11

Component 2:  0 1 2 3 4

Component 3:  5 6 7
```

Print a maximum of 16 vertices on each line before skipping to a new line; in case there are more than 16 vertices in a component: +3 point bonus for indenting the second line so that vertex numbers line up vertex numbers on the previous line and using fixed field widths (width 4) so that everything lines up nicely.

Skip a blank line between components as illustrated in the example. There is no requirement regarding the order in which the components are numbered.