The University of North Carolina at Greensboro
CSC 330: Advanced Data Structures
Prof. Stephen R. Tate

Handout 10
November 8, 2016

Assignment 5 – Due Thursday, November 17

Objective: The objective of this assignment is for you to gain experience working with the basic graph data structure, including iterating through vertices and adjacencies. This assignment is not as long or intricate as other full-credit assignments, so take the time to experiment with the graph data structure and get comfortable working with it. The time you invest in this will pay off on the final assignment!

Background: Courses in Computer Science depend heavily on prerequisites, and it is common to represent prerequisites as a graph (like on the prerequisite chart we publish on our department web page). It is easy to create a graph of courses linked to prerequisites by scraping data from the *UNCG Bulletin*, and you can use that as input to construct a graph with an edge from a course to each of its prerequisites. However, sometimes we might want to look at this graph in the other way: For each course that acts as a prerequisite, what courses depend on it? This is the *transpose* of the original graph, or the graph with the same vertices and connections, but with the direction of each edge reversed. Your goal for this assignment is to write methods that will allow you to take a graph of Computer Science courses and prerequisites, and print the transpose graph. Samples of the output for both the original graph and the transposed graph are given on the following pages.

What To Do: Start with the code in Bitbucket, as in previous assignments: fork the "Assign5" repository, rename it to include your username, grant read access to the class administrators, and then use NetBeans on your computer to clone it so you can work with it.

You are to write two methods (stubs are given in the provided code):

- `printGraph()` prints the graph in a readable form, as shown in the samples on the following pages. Note that the output is "nice" in the sense that each vertex is followed by an appropriate, grammatically correct phrase on the same line, depending on whether there are zero, one, or more adjacent vertices. Note also that the vertices are given in sorted order — you actually get that "for free" if you change one data structure that is used in the Weiss graph implementation. These small touches matter!
- `getTransposedGraph()` operates on a graph and returns a new graph that is the transpose of the original graph (which is not changed).

Submission Instructions: Using NetBeans, commit all changes to your project and do a "push to upstream" to put the most up-to-date files on the Bitbucket server. Remember: Do *not* create a pull request — I will clone your repository (if it exists and you granted me access) at 12:30 on the due date, and will assume that is your submission. If you intend to keep working on your project and submit late, please let me know by email, and I will ignore your repository until the late submission deadline.

Original Graph Output: This shows the output of the `printGraph()` method when called on the original graph (courses with edges to their prerequisites). *Your output will not be two columns, of course — it's just printed that way here to save space.*

```
CSC130 -- no edges out
CSC230 -- edge out to:
    CSC130
CSC250 -- edge out to:
    CSC130
CSC261 -- edges out to:
    CSC230
    CSC250
CSC312 -- edges out to:
    CSC230
    CSC250
CSC330 -- edges out to:
    CSC230
    CSC250
CSC339 -- edge out to:
    CSC330
CSC340 -- edge out to:
    CSC330
CSC350 -- edge out to:
    CSC250
CSC463 -- edges out to:
    CSC562
    CSC567
CSC464 -- edge out to:
    CSC463
CSC465 -- edge out to:
    CSC464
CSC471 -- edge out to:
    CSC330
CSC510 -- edges out to:
    CSC330
    CSC567
CSC521 -- edges out to:
    CSC340
    CSC350
CSC522 -- edges out to:
    CSC330
    CSC350
```

```
CSC523 -- edges out to:
    CSC130
    CSC350
CSC524 -- edge out to:
    CSC523
CSC529 -- edges out to:
    CSC330
    CSC350
CSC539 -- edges out to:
    CSC261
    CSC330
CSC540 -- edge out to:
    CSC340
CSC553 -- edge out to:
    CSC350
CSC555 -- edge out to:
    CSC330
CSC561 -- edges out to:
    CSC261
    CSC330
    CSC350
CSC562 -- edges out to:
    CSC261
    CSC340
CSC567 -- edges out to:
    CSC261
    CSC330
CSC568 -- edges out to:
    CSC330
    CSC567
CSC580 -- edges out to:
    CSC330
    CSC350
CSC583 -- edges out to:
    CSC567
    CSC580
```

Transposed Graph Output: This shows the output of the transposed graph (the graph with all edge directions reversed). From this output, we can easily check a course to see what later courses depend on it. Note how important this class (CSC 330) is!

```
CSC130 -- edges out to:
    CSC230
    CSC250
    CSC523
CSC230 -- edges out to:
    CSC261
    CSC312
    CSC330
CSC250 -- edges out to:
    CSC261
    CSC312
    CSC330
    CSC350
CSC261 -- edges out to:
    CSC539
    CSC561
    CSC562
    CSC567
CSC312 -- no edges out
CSC330 -- edges out to:
    CSC339
    CSC340
    CSC471
    CSC510
    CSC522
    CSC529
    CSC539
    CSC555
    CSC561
    CSC567
    CSC568
    CSC580
CSC339 -- no edges out
CSC340 -- edges out to:
    CSC521
    CSC540
    CSC562
CSC350 -- edges out to:
    CSC521
    CSC522
    CSC523
    CSC529
    CSC553
    CSC561
    CSC580
CSC463 -- edge out to:
    CSC464
CSC464 -- edge out to:
    CSC465
CSC465 -- no edges out
CSC471 -- no edges out
CSC510 -- no edges out
CSC521 -- no edges out
CSC522 -- no edges out
CSC523 -- edge out to:
    CSC524
CSC524 -- no edges out
CSC529 -- no edges out
CSC539 -- no edges out
CSC540 -- no edges out
CSC553 -- no edges out
CSC555 -- no edges out
CSC561 -- no edges out
CSC562 -- edge out to:
    CSC463
CSC567 -- edges out to:
    CSC463
    CSC510
    CSC568
    CSC583
CSC568 -- no edges out
CSC580 -- edge out to:
    CSC583
CSC583 -- no edges out
```