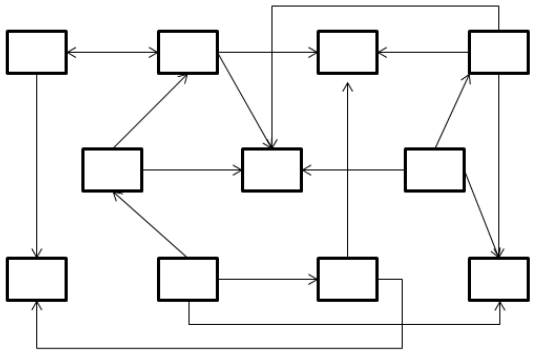
hw-8 CS230 40 points Due: 4/14/14

We have a problem. We want to hangout with our friends, but we also need to find a way home. So, we will create a graph of all establishments in town that you may venture out to. This graph will contain the names of the establishments and the cost of traversing amongst them (to keep things tractable we will limit the number of paths between establishments).

Part 1:
Your code will create a graph similar to that below. Note that each node is an establishment. The arcs connecting establishments will be the weighted paths you are allowed to take.



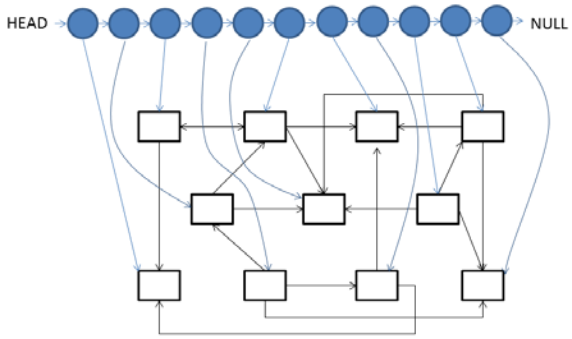
To further simplify the creation of the graph, the data file (hw8.data) will consist of three parts: a list of establishments (one to a line) and a list of arcs. For example:

Applebees
GroundRound
BlueMoose
DrunkenNoodle
Home
STOP
Applebees BlueMoose 10
Applebees DrunkenNoodle 13
GroundRound Applebees 2
GroundRound DrunkenNoodle 7
GroundRound Home 52
STOP STOP 0
GroundRound

Notes:

- Establishments with multi-word names will be truncated into a single word (e.g. Ground Round -> GroundRound). So each line will have 1 word with a maximum character length of 100.
- The word STOP will indicate the end a section.
- The list of arcs will contain 2 words per line and an integer. First will be the start-point of the arc, followed by the end point, followed by the cost/weight. All arcs will be one-direction (where the diagram shows bidirection it is actually 2 one-direction). The maximum number of arcs possible from any node is 10. But, not all nodes will have 10 outgoing arcs.
- The words STOP STOP 0 will indicate the end this section (to be compatible with the rest of this section).
- The last section will be the name of one of the establishments. This will be the starting point for our journey home. Note that “Home” will also be a node in the graph.

To further simplify the creation of the graph, I suggest that you use (modify) your linked list code to build up an index. Trust me this will make the rest of the coding MUCH easier. For example, the blue circles are your linked-list code modified such that the payload contains the name of the establishment and a pointer for that establishment into the graph. You construct the linked-list and the nodes of the graph with the first part of the data file and then use the linked-list to find the nodes in the graph such that you can create the pointer references (arcs).



Part 2:

Take me Home One-Direction graph! Given a starting point for our journey home, implement a random walk algorithm (aka drunkard's walk). Given a node, the algorithm randomly selects the next path to take. Repeat until you get home. Accumulate the costs along the way. I suggest you implement this part as a separate function (just a hint).

Finally, your program must display the starting node, the ending node, and the accumulated cost.

EXTRA CREDIT:
1. (5 points) In addition to the drunkard's walk, also implement a greedy method and display the starting node, the ending node, and the accumulated cost for that method.

REQUIREMENTS:
1. Your program must run in Streibel 115/109 or on shell.aero.und.edu.
2. Your full name must appear as a comment at the beginning of your program.
3. Your source code must be named hw8-yourname.c
4. Email your source (subject hw8-yourname) to rmarsh@cs.und.edu
5. Note in your email if you did the extra credit and to what level.