ETEC2101	Lab10: **Graph Algorithms**	Points: **50**

1. Work from a Lab9 solution that can load and display a graph
2. Add this functionality that works with our *visual graph (the one from the text file)*
    a. Very important: I don't want the result of a traversal to be "embedded" in the Graph or TextCircles – make sure we can easily create / throw away traversals by using the SearchHistory technique discussed in the slides
    b. (**5 points**) Allow the user to select a node to start the traversal – visualize this somehow (I changed the outline color)
    c. (**15 points**) Compute a breadth-first traversal of all nodes that can be reached from the starting node.
    d. (**15 points**) Compute a depth-first traversal of all nodes that can be reached from the starting node. (+5 points) if you do this without recursion (probably using stacks)
    e. (**15 points**) "Show" the result of the traversals somehow. If you do item 3, you can earn these points that way. A simpler approach (to just get these points) might be to print out the search history using cout.
3. (**15 points**) Visualize (using SFML tools) the search history of BFS and DFS [this meets the requirements for 2e as well]
4. (**5 points**) Have a way to change which traversal algorithm you're using at run-time (I used the F1, F2, and F3 keys)
5. (**25 points**) A* with an open *heap* and show the actual selected path (in addition to the search history)
6. (**15 points**) Instead of showing the solution all at once, allow the user the option of stepping through (include on-screen prompts showing how to do this). It's OK if the solution is pre-computed and you're just showing a part at a time
7. (**5 points**) Generate [and look at the index.html file in a browser] the doxygen documentation for our now complete ssuds project.
8. Here's a video of my solution: **https://youtu.be/F-LXMEsxN2w**