ETEC2101	Lab6: **Inheritance (and SFML)**	Points: **100**

1. (**30 points**) Create a ssuds::Stack and Queue class (both in their own header file) that leverage our LinkedList class. For one, use a "has-a" implementation and for the other use a "is-a" implementation. For both:
    a. include an iterator. You almost surely do *not* need to write your own – just use the iterator we already have in LinkedList.
    b. Have only one way to add/remove (stacks=push/pop; queues=enqueue/dequeue). Both of these should remove the thing and return it.
    c. Have a size method
    d. Have an "empty" method that returns true if this is empty.
    e. Any other methods you choose to expose (but do not give the user the tools to "break" the rule of the data structure)
2. (**10 points**) Google tests of the above

[A software engineering problem that uses Stacks and Queues. I chose not to put these classes in ssuds]

1. (**10 points**) Incorporate SFML dependency into our project (both Debug and Release builds). Ask on discord or in class if you have any questions on the use of SFML not covered by the tutorials I've linked in the slides (feel free to google SFML techniques as well!)
2. (**20 points**) Make a TextCircle class that inherits from sf::Drawable[1] and overrides the draw method to draw centered text on top of a circle that just encircles the text. Have a method that returns true is a sf::Vector2f is within the circle (use Pythagorean Theorem for this – I can help with the math if you ask). Have any getters / setters you think appropriate (most will simply call a member / inherited method). Do the full .h/.cpp split since this is not a templated class.
3. (**40 points**) Make a WordDrawer class (again, strictly adhere to the .h/.cpp split!) that has the following behavior [*yes, / do want the control-scheme I have described here* 😊 *]*
    a. Loads and keeps open a *binary* ifstream file connected to a SCOWL[2] *text file* (I would recommend the American-words.80 file). I do NOT want you to read the entire file into memory. See the next item.
    b. Maintains logic to store 10 random[3] items in a QUEUE (this can be strings or TextCircle's – your choice). If the last item is served up, get another 10 items from the file and store them in the QUEUE.
    c. If the user presses the left mouse button on an empty area of the screen (see next item) and there is not an active word, dequeue one from your QUEUE and make it move with the mouse. If the user left clicks again, make it "stick" to the screen. If the user right-clicks while the word is moving with the mouse, discard it.
    d. Show status text on-screen to indicate what the user should do.
    e. (**+10 points**) If the user left-clicks on an existing circle (make sure the top-most one is picked first if there is more than one), allow the user to move it as long as the left-mouse button is held. Once the mouse button is released, place the word at that spot (but preserve the "layer" it was at when originally placed). Make sure to preserve the offset between where the circle was first clicked and apply that as its moved so the circle doesn't "hop" when clicking on it.
4. (**10 points**) Maintain a STACK (either in the WordDrawer class or elsewhere) that allows us to press CTRL-Z to undo the last change (placement of a circle and/or movement of a circle [if doing item 5d)
    a. Display the head word in the QUEUE as a TextCircle that moves with the mouse. Once the user clicks, "place" that item into a STACK (again, of text or TextCircle's).
5. Here's a video of how I'd like yours to look: https://youtu.be/dvSnPBu9Y9g

---

[1] I had initially wanted to have you inherit from sf::Circle, but the sf::Shape class (parent of sf::Circle) declares its draw method, which we need to override, as private. This forces us to use a hybrid has-a/is-a implementation where we have a Circle object internally, but we use inheritance to inherit from Drawable. If you understand this, you have a good handle on inheritance (for the final exam).
[2] Spell-Checker Oriented Word List
[3] As in: each time you get one word, you find a random spot in the file, and return that word. Ask Jason for the technique to find a random word in a text file opened in binary mode if he hasn't already volunteered this.