2.1 Written Report

1. **Keep track of how much time you spend designing, coding and correcting errors, and how many errors you need to correct**
2. **Keep a log of what proportion of your errors come from design errors and what proportion from coding/implementation errors.**

3 March

Assignment was given, did not start. Basic files with no methods were made to begin the assignment.

Some classes were taken directly from SENG110 assignment as a template to creating the class. Most of

the classes were also made with basic functions.

11 March

**Error**: main file could not read text input file, so I resorted to having a class to implementing the steps to reading. Mainly tested during the day. Other methods were made in the mean time by concept and paper drawings of how I assume the program would run (most of which would be heavily modified in the future).

11 March (Night)

Attempted to make points as arrays as it would be easier to feed an array instead of making complicated loops to be added into each polygon. Both ways of inputting points into a polygon object were acknowledged but the arrays was easier to understand when drawing on paper.

**Error**: point needed input parameters for number of elements to be made, but the point class was not

made to have parameters for number of points. This concept was difficult to understand and a lot of

testing was required.

Also made progress on MyPolygons class as a Linked List (mainly append() and prepend()).

ComparePoly was renamed to ComesBefore as a classmate suggested it was easier to understand when programming the insert() method later.

toString() method was made in all files.

**Error**: polygon needed to be fixed as it was difficult to figure out how to loop through each point.

12 March

Changed constructor to implement polygon objects.

Prepend() and append() functions are complete and should be working.

Added a next() method to MyPolygons to iterate to the node after current.

removeFromHead() is added but not tested properly.

setCurrent() was removed as it is not required as a circular doubly linked list.   
Changed constructor to only have 1 parameter as it became known that initialNext and initialPrevious was not required in a doubly linked list.

Attempted to changed point objects to be instantiated as an array but still had the same error.

Added int position to Node class to easier tell position of node being added. setPosition() and getPosition() as it was thought it would serve useful later (it didn’t).

14 March

A new file was released on Blackboard which would serve as an example as the “toughest” text file they would test on the program.

**Error**: my splitting text file based on Enter was not valid for this new file. Therefore, a new polygon would have to be made for each “P” the file scanner recognizes. Dan explained the Java functions I should be using in my lab which helped me solve this error the following day.

15 March

A new polygon could now be used to make a new polygon, and the file can also store the int variable which would be used in the polygon constructor to tell the program the number of elements in the point object arrays. Points could be made easily and stored with their respective X and Y coordinates.

**Error**: Tried to add polygons and points in each polygon but am stuck. A drawing was used to understand the adding of points into polygons. Basically, it would run through each current and sentinel node and make sure the order and conditions in which nodes were added were being added correctly.

17 March

Almost all of my methods and Nodes created were either removed or changed because of Dan’s explanation of how to work a circular doubly linked list. The idea was essentially that only 2 currents are to be made in the linked list (current and sentinel) and the concept made sense on paper. However, it is more difficult to code especially when there are no references to use. The remaking of the MyPolygon class would take the most time.

By night, the file scanner was fully functional and even worked on Dan’s complicated text file.

18 March

Points are being put into polygon object BUT (**Error**) area function cannot recognise polygon.The fix was that getY() and getX() on point had to be made to iterate through each point based on the shoelace formula.

Polygon class can now be run with the file scanner in main class (previously, only an integer was set to indicate number of polygons based on number of “P”s and number of points on int after “P”).

All polygon functions being tested in main class has now been shifted into polygon function to now work with point class directly (keeping encapsulation).

Error: distanceFromOrigin() and append() was removed and remade respectively as it would not function properly.

File reader class is deleted from program as it is no longer needed.

19 March – 21 March

**Error**: Nodes could be sorted in terms of area. 3 days were dedicated to getting the insertionSort() method to work. This method is connected to multiple methods. Therefore, insertion(), prepend(), append(), comesBefore() and remove() were modified intensely and reset(), getCurrent(), getSentinel(), getPosition() and getLength() were added during this process. Long periods of slow debugging and drawings were made to understand and solve node insertion errors to sort the list. Therefore, Visual Studio Code was used instead of Sublime 3. My strategy to inserting the list changed multiple times but I settled with created another empty linked list and adding from the previous unsorted list into the new list which would be sorted.

**Error**: accounting for 2 and 3 decimal places for output. This was a simple browsing of Stack Overflow pages and implementing it in the toString() methods.

**Error**: comesBefore() also needed an extra if guard for when 2 areas are within 0.1% of each other, resorting to the polygon with the point closer to the origin as the smaller area.

1. **Given what we have covered in Topic 3 (Inheritance), how could you treat Rectangles and Squares as special cases of this assignment?**