Andrew Nguyen AXN210010
Project 4 Pseudocode

Test Cases
1. Test for trig functions being derived properly with correct exponents and signs
2. Check for chain rule being applied for trig functions
3. Check if the outer coefficient is 1 when there is no actual outer coefficient written
4. Check if a polynomial term is derived correctly
5. Check if the + and - signs are captured correctly through indexing
6. Check if the trig functions can be found properly when parsing
7. Check if the outer coefficient is correctly selected
8. Check if the inner coefficient is correctly selected in terms with trig
9. Check if the exponent is correctly selected
10. Check if exponent is 1 if there is no ^ being used in the term
11. See if the linked list is sorted properly
12. See if the linked list prints the nodes properly
13. See if a node with a trig identifier is printed properly
14. See if terms such as x^0 and x^1 are simplified in the display
15. See if operators do not appear at the very front of the equation

```
void Derive term (Node term)
	Check if there is a trig identifier
		Switch case
			Convert sin(x) to cos(x)
			Convert cos(x) to -sin(x)
			Convert tan(x) to sec^2(x)
			Convert csc(x) to -csc(x) cot(x)
			Convert sec(x) to sec(x) tan(x)
			Convert cot(x) to -csc^2(x)

		For deriving tan and cot, set the exponent to 2
		For deriving cot, csc, and cot, multiply the outer coefficient by -1
		For deriving csc and sec, use a special trig identifier

		Check if there is an inner coefficient and apply chain rule
			Multiply the term's outer coefficient by the inner coefficient
	else
		Multiply the outer coefficient by the exponent
		Subtract the exponent by 1
	end
end
```

```
Node Parse term (String termText)
	Create a new Node

	Find occurences of "sin", "cos", "tan", "csc", "sec", and "cot"
	Find occurence of 'x'
	Find occurence of '+' and '-'

	If '-' is near the beginning of the term
		Multiply the outer coefficient by -1

	If there are occurences of trig functions
		Set the Node's trig identifier to the appropriate trig function found
		Look to the index left of the 'x' index and check if there is a number
			Get the number in between the trig index + the remaining characters of the trig function
			and the 'x' index

			Set that number to the inner coefficient
		end
	else
		If 'x' cannot be found
			Set exponent to 0
			Set outer coefficient to the termText converted to int (using stoi)
		else
			Look for occurence of '^'
			If there is '^'
				Get number between '^' and the end of the string and set that to the Node's exponent
			else
				Set exponent to 1
			end

			Set outer coefficient to number between the '+'/'-' indexes and the 'x' index
			If outer coefficient does not exist
				Set outer coefficient to 1
		end
	end

	Return the Node
end
```

```
void Read file (String file name)
	Check if file exists and can be opened

	Create line string variable
	While getting each line of the file
		Check if the line is not empty
			Create a new LList

			Create a string called termText
			While the length of the line is not zero
				Find indexes of first occurence of '+' or '-' in the line
				Get the lower index of the two

				If index is not the last index
					Split line into two strings using the index
					term = line.substr(0, index)
					line = line.substr(index + 1)
				else
					termText = line
					line = ""
				end

				Call the parse term function using termText
				Add the Node given from the function to the LList
			end

			If the length of the LList is not zero
				Sort the LList
				For each node in the LList
					Derive the node by calling the derive function
				end
				Display the LList
			end

			Delete the LList
		end
	end
end


int Main function ()
	Take file name from user input
	Parse the file from the file name
end
```