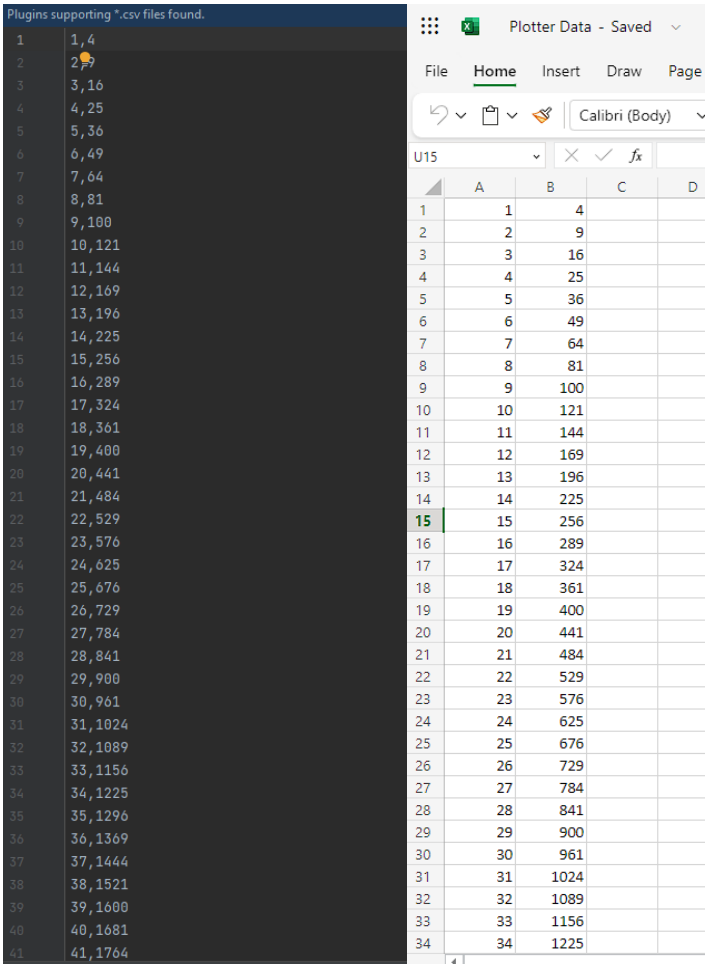
Java Plotter Salter Smoother Report

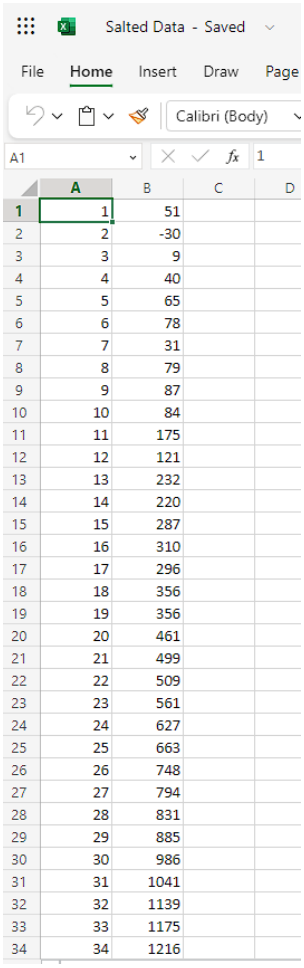
For the first part of our coding assignments apart of project 2, we had to write some code in java that dealt with a function of our choice. The function I chose to work with is y = $x^2 + 2x + 1$. The first part of the code was to figure out a method that would plot the x and y outputs of said function into a CSV file, for a range of about 1-100 x values. So, using the generateOriginalCSV method, along with some imports to hand the output to a CSV file, I wrote a for loop that went from 1 to 100 and then output these data points into file named original.csv. Below is what that csv looks like from the IDE.



This CSV goes all the way up to 100 values which the screenshot cannot physically include. I then took this CSV, downloaded it to my computer, and imported it into an excel file, as can also be seen above.

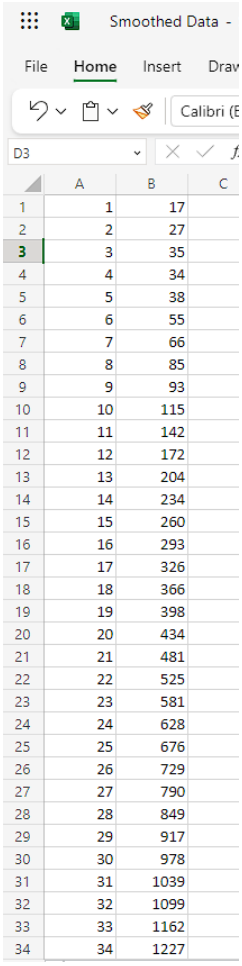
The next part of the assignment was to add a method that would “salt” the previously plotted data. This means there needed to be some code that would accept the previously plotted data in the CSV, read it, and then loop through the y values and add or subtract a random number from it. The range that I configured for this random number was from –50 to 50. Meaning the largest number it could subtract was 50 and the largest number it could add was 50. I was able to do this by using the BufferReader import and the random import which set the bounds of the range I was looking for. The method then

created a new CSV file named “salted.csv”, with these new y values. Below is the screenshot of the CSV values from excel after I downloaded the file and imported it there.



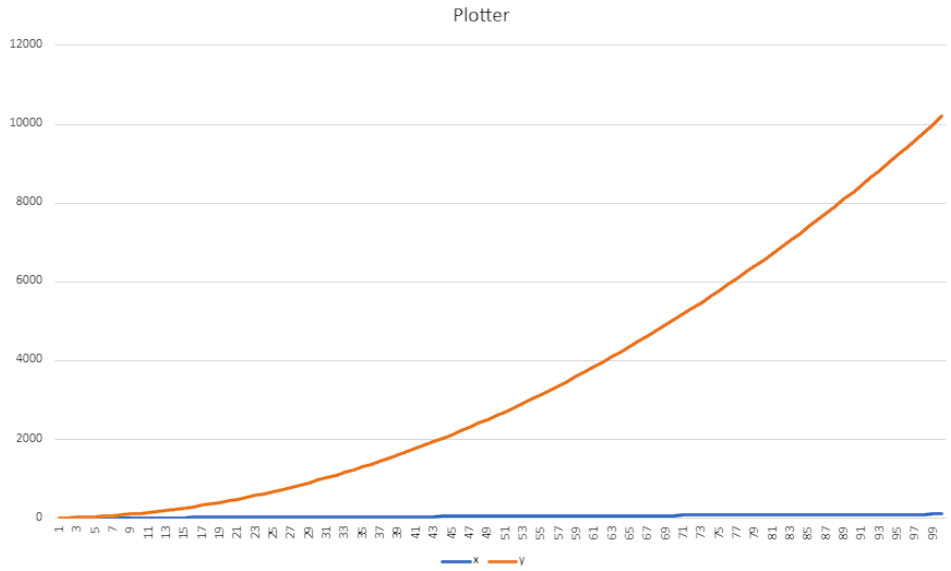
	A	B	C	D
1	1	51		
2	2	-30		
3	3	9		
4	4	40		
5	5	65		
6	6	78		
7	7	31		
8	8	79		
9	9	87		
10	10	84		
11	11	175		
12	12	121		
13	13	232		
14	14	220		
15	15	287		
16	16	310		
17	17	296		
18	18	356		
19	19	356		
20	20	461		
21	21	499		
22	22	509		
23	23	561		
24	24	627		
25	25	663		
26	26	748		
27	27	794		
28	28	831		
29	29	885		
30	30	986		
31	31	1041		
32	32	1139		
33	33	1175		
34	34	1216		

The final part for this coding assignment was to add a method that now “smoothed” the previously salted data. So, similarly to the salted data method, this one would read in that data, and then smooth it by looping through the y values and replacing them with the average of the 3 y values to the left and right of it. This method was a little bit more complicated, but I used the same style as before to read in the salted data, and then I created a for loop which handled the actual smoothing part where it found the values to the 3 left and right, took the average, and replaced the value with that number. I ensured that the code worked properly by completing the math myself on the first few numbers and it did come out right. Below is a screenshot of how these values look in excel after I took the file and opened it in excel.

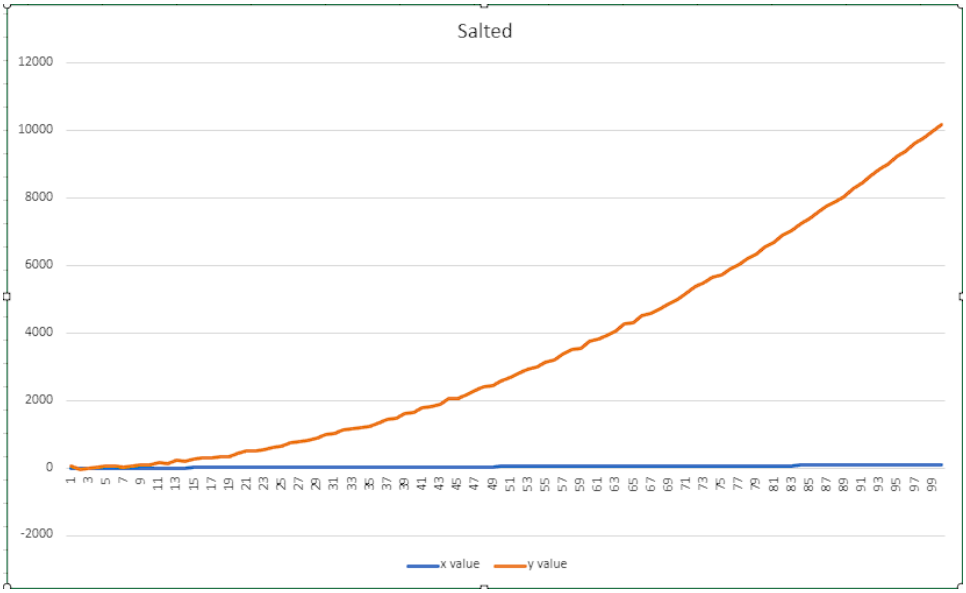


	A	B
1	1	17
2	2	27
3	3	35
4	4	34
5	5	38
6	6	55
7	7	66
8	8	85
9	9	93
10	10	115
11	11	142
12	12	172
13	13	204
14	14	234
15	15	260
16	16	293
17	17	326
18	18	366
19	19	398
20	20	434
21	21	481
22	22	525
23	23	581
24	24	628
25	25	676
26	26	729
27	27	790
28	28	849
29	29	917
30	30	978
31	31	1039
32	32	1099
33	33	1162
34	34	1227

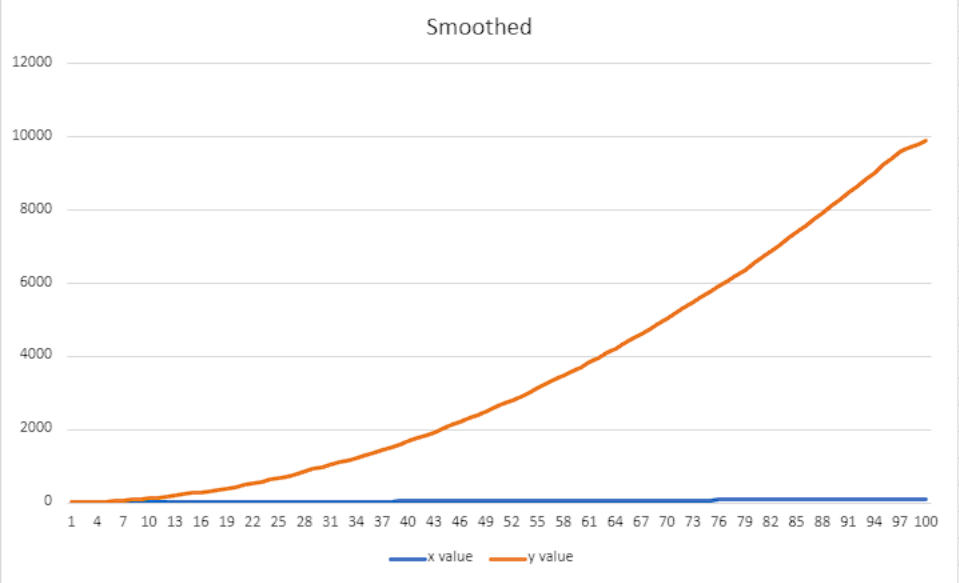
Then, for the final part of the assignment, I made line graphs in excel for each form of data, the screenshots of said charts are below.



This is the original graph for the normal x and y values calculated by the program and output into the file. As you can see, it is a normal curved line graph that a basic polynomial function like the one I chose normally would show. The x values, noted in blue along the x axis, go from 1 to 100, and the y values, noted along the y axis, go all the way up to 12,000 although the actual values only go to 10,201.



This is the graph of the salted data, and as you can see, salting the data made the line for the graph very bumpy. Since each value can be added or subtracted by 50, that affects the lower numbers, and it is clear how bumpy the line is towards the beginning. Despite this, the line maintains its normal curve like the normal plotted data.



Lastly, the smoothed data does not really convey much for the function that I chose. The only things that stand out to me about this graph are the very slight bumps which are much less noticeable compared to the salted graph. Also, the highest value is 9,897, which is different than the other two whose highest value both exceeded 10,000.