2190101 Final Project

1. Objectives of final project

- Be able to apply knowledge from the Java programming class to solve the project.
- Raise public awareness of importance and potential consequences of national emergency policies on the Covid-19 pandemic situation in Thailand and other countries.

2. Introduction

In this final group project, you will study the progression of the global Covid-19 pandemic. In particular, your group will write Java codes and a very brief report to process time series of cumulative confirmed cases in different countries and make projections based on certain scenarios, and submit only one copy of your work for the group.

This final project serves to replace an in-class final exam, which we could not have, so different groups are not supposed to collaborate in any manner. Detected plagiarism across groups will be severely punished, and possibly escalated to the university administration. The deadline for this final project will be on Sunday 3 May 2020 at 9:00AM.

3. Data source

You should find the relevant data set in the `time_series_covid19_confirmed_global.csv` file

This file contains the numbers of cumulative confirmed Covid-19 cases in different countries on every day starting from 22 January 2020 to the current date (updated every day). In particular, each row of the csv file contains the time series of cumulative confirmed cases reported at a particular area. Download and open the file to see what information it contains. Notice that for some countries, the data for many places (Provinces/States) in each of these countries could be distributed across many rows.

3.1. Data preprocessing

Your first job is to write a Java method `getInfected()` in your main program to properly read the csv file, extract the time series of confirmed cases for suitable countries, store them in appropriate data structure and return it. To this end, your codes must:

- Create a new class `CountryInfected` (in a separate file `CountryInfected.java`) representing the time series for one country with private data members being:
    - `String country:` for the name of the country.
    - `int [] infected:` for the array of the numbers of confirmed cases on different dates starting from 22 January 2020 (`infected[0]`) till the date that you execute the code.
- Your method `getInfected()` must extract the time series of confirmed cases for all suitable countries (as explained below) from the csv file, store them into an array of objects of class-typed `CountryInfected` and return it. In particular, the method should have the following header:

```
public static CountryInfected [] getInfected()
```

Your `getInfected() method` must successfully perform all of the following.

A. Some country names in the csv file are not appropriate; these are “Holy See”, “Korea, South” and “Taiwan*”. Your method `getInfected()` is supposed to change them to “Vatican City”, “South Korea” and “Taiwan”, respectively, before storing the changed names in the data member `country` of each element of the returned array. To summarize, below is the table for current country names to be changed by your program and the target names:

Current names in the csv file	Names to which your program will change
Holy See	Vatican City
Korea South	South Korea
Taiwan*	Taiwan

B. As we mentioned that the data for some countries could consist of time series data in different places

(Provinces/States) these countries spreaded out across many rows of the csv file. For every such country, your method `getInfected()` must aggregate all those data across the different rows in the csv file into a single time series for that country.

C. Your method `getInfected()` is supposed to disregard all countries whose numbers of confirmed cases have never gone above 100.

In summary, your correct implementation of `getInfected()`should return an array of class-type `CountryInfected` whose size equals to the size of all countries with cumulative cases up to the current date more than 100.

4. Prediction models

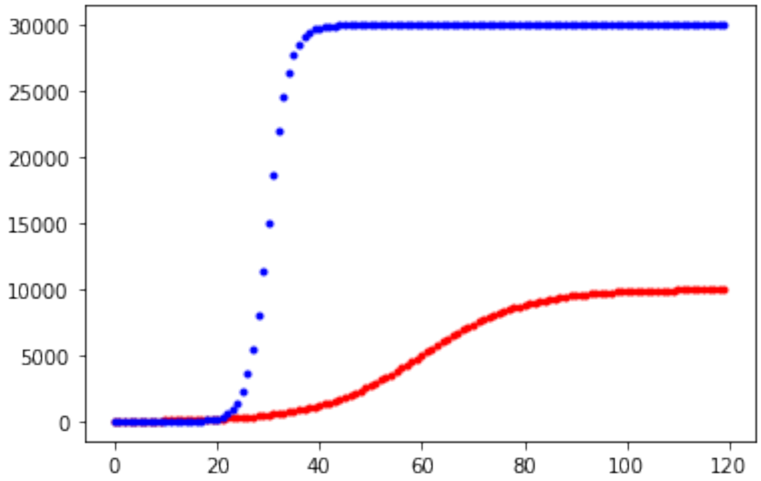
We shall use very simple prediction models for the two scenarios

- *S-curve model:* This model is assumed to be the result of proper interventions from governmental mandated policies such as closure of schools, universities, restaurants and department stores, work-from-home order and encouraging social distancing. In particular, the model consists of the non-negative parameters: S, D, L, M and projects the number cases $s(d)$ on day d according to the following formula:

$$s(d) \; = \; S + \frac{M}{1 + e^{-L(d-D)}}$$

Basically, this function models the progression of the controlled epidemic (number of confirmed cases) starting off from S cases at time $d = -\infty$, increasing sharply to the center of the S-curve shape to $S + \frac{M}{2}$ at day $d = D$ and then saturating off while still increasing to the final value of $S + M$. The parameter L determines the speed of approaching and diverging from the center transition point. The higher it is, the more abrupt the curve transitions past the mid-value of $S + \frac{M}{2}$.

Sample plot for S curves are as shown below:



The blue plot is an S curve with the parameters: $S = 10, D = 30, L = 0.5, M = 30{,}000$, and the red plot with the parameters: $S = 10, D = 60, L = 0.1, M = 10{,}000$.

The underlying assumption for this model is that such interventions will slow the progression of the epidemic down by reducing the slope of the curve after passing the mid-point time. You can visit the sites below for more information about S-curve functions:

- https://en.wikipedia.org/wiki/Sigmoid_function
- https://stats.areppim.com/glossaire/scurve_def.htm
- A paper from sciencedirect.com (copy and paste hyper-link if a click does not work)

- *Do-nothing model:* This model assumes that there are no such interventions (at the beginning of the outbreak). It can also be used to model the situation after such interventions are repealed. So basically, in this situation, we assume that the pandemic will follow its natural course. We model this natural course by letting the new number of infected cases be determined by the average ratios of numbers on consecutive days over the last four days. In particular, the number of infected cases $n(d)$ on day d could be computed as

$$n(d) = \frac{\frac{n(d-1)}{n(d-2)} + \frac{n(d-2)}{n(d-3)} + \frac{n(d-3)}{n(d-4)}}{3}$$

Your job is to write two Java methods `getSCurve()` and `getDoNothingCurve()` to calculate these projected trends in the two scenarios for the specified number of future days from past data. In particular, the methods should have the following headers:

```
public static double[] getSCurve(int[] pastData,
   int numFutureDays, double[] paramLowerBounds,
   double[] paramUpperBounds)

public static double[] getDoNothingCurve(
   int[] pastData, int numFutureDays)
```

The second method is straightforward, whereas the first requires some further explanation.

In particular, your method getSCurve() should strive to find the parameters S, D, L, M that will best suit the past data. For example, you can write four for loops iterating over different values of each of these parameters starting from the paramLowerBounds[i] to paramUpperBounds[i], where i = 0, 1, 2, 3, for S, D, L, M, respectively. In every iteration, you could compute an error, .e.g., the mean square error (MSE), between the resulting S-curve and real data on certain past dates and select the parameter set that will yield the lowest error. However, you are free to do this fitting by other means that you deem best. Notice that your projection based on the S-curve model will be evaluated based on the error from the real future data. For example, we can evaluate you 10 days after the project is due, and your score in this part will depend on the error between your projection and the real on those 10 days. Note that the MSE is defined as $MSE = \frac{\sum_{d\ in\ \Lambda}(s(d)-r(d))^2}{|\Lambda|}$,where $s(d), r(d), d\ in\ \Lambda$ are your projected and real values, respectively, for all days in the interval Λ with $|\Lambda|$ number of days.

To improve your confidence on your S-curve project, you can test the predictive power of your fitted model by, for example, splitting the past data into two sub-intervals one before the other. For example, your first sub-interval could start from the beginning of 22 January 2020 up to a certain date before your submission date, and the second sub-interval could start from that date to your submission date. You can use only the data in the first older sub-interval to fit (train) your S-curve model as if the model has not seen the data in the second sub-interval. Then, you could test its predictive power by computing the “test” error based on the data in the second sub-interval, which you presume to be “in-the-future” in the eye of your model. If your model predicts reasonably well, the test error should be small enough for you to be confident that it will predict well into the real future.

Note that your `getSCurve(), getDoNothingCurve()` methods will return an array of type `double` whose length equals to `numFutureDays`. To let us see the parameters of the fitted S-curve model producing the projection from the `getSCurve()` method, you must printout the model parameters out before your return from the method using the following message format:

```
“The fitted S-curve model has S=[OptS], D=[OptD],
L=[OptL], M=[OptM], with the first projected day being
d=[firstProjDay].”
```

In the above `OptS, OptD, OptL, OptM` are the parameters your algorithm computed for the invocation of the `getSCurve()` method and `firstProjDay` is the value d that you use (in the S-curve formula above) to compute the first future day out of `firstProjDay` days. For example, if you run the code on 30 April 2020 and let d be 1 for the first day in the csv file: 22 Jan 2020, your `firstProjDay` will be = `10+28+31+30+1 = 100`, but you could start on 22 Jan 2020 with any d you like.

Lastly, In addition to your method `getSCurve()` being evaluated on its accuracy, it must also be sufficiently time-efficient. In particular, to get a full credit on its time-efficiency (see, Section 7 Grading Criteria below), it must not take longer than 10 minutes in total to finish.

5. Producing your projections for different countries

Now to conclude the project, you must write a test code (in the `main()` method) to utilize all your previous codes. Depending on your group number, your main method should extract past data of the following countries and produce 90-day projections based on the S-curve and Do-nothing models.

Group	Countries		
1, 11	United States	Czechia	Thailand
2, 12	Spain	Australia	Thailand
3, 13	Italy	Serbia	Thailand
4, 14	France	Dominican Republic	Thailand
5, 15	Germany	Panama	Thailand
6, 16	United Kingdom	Bangladesh	Thailand
7, 17	Turkey	Malaysia	Thailand
8, 18	Iran	Columbia	Thailand
9, 19	Russia	South Africa	Thailand
10, 20	Brazil	Egypt	Thailand

Your main program should produce a csv file containing the projections for the three assigned countries. In particular, each csv file should be named X_Y.csv, where X is your section Y is your group number (you can find your group number in your group names in myCourseVille) and should consist of six rows containing the projections based on the S-curve model and Do-nothing models for the first, second and third countries respectively in that order. In particular, the csv file for Group 1 should look like

```
su1,su2,su3,...,su10
nu1,nu2,nu3,...,nu10
sc1,sc2,sc3,...,sc10
nc1,nc2,nc3,...,nc10
st1,st2,st3,...,st10
nt1,nt2,nt3,...,nt10
```

The `su1,su2,su3,…,su10` and the `nu1,nu2,nu3,…, nu10` are the accumulated confirmed cases of the United States, projected 10-day ahead by the S-curve and the Do-nothing models, respectively. The `sc1,sc2,sc3,…,sc10` and the `nc1,nc2,nc3,…, nc10` are the accumulated confirmed cases of Czechia, projected 90-day ahead by the S-curve and the Do-nothing models, respectively. The `st1,st2,st3,…,st10` and the `nt1,nt2,nt3,…, nt10` are the accumulated confirmed cases of Thailand, projected 10-day ahead by the S-curve and the Do-nothing models, respectively.

Notice that your projection in the csv file will be evaluated based on the MSE of your projected data and real data. *In other words, your group will have to COMPETE WITH ALL OTHER GROUPS to get the LOWEST MSE to get the BEST grades.*

6. What must you submit?

- A correct implementation of class `CountryInfected` in the `CountryInfected.java` file.
- The main program titled `Proj2190101_X_Y.java`, where X is your section and Y is your group number which contains the correct implementations of `getInfected(), getSCurve(), getDoNothingCurve()` and `main()` methods as explained above.
- A csv file titled X_Y.csv, having 6 rows, containing the S-curve and Do-nothing projections of the three countries assigned to your group as in the table above.
- A very brief report explaining your method for S-curve projection in onl the PDF form titled `SCurve_X_Y.PDF.` The report must not be longer than a page with a standard margin, using the Thai Sarabun PSK font of size 14 points.
- Put everything in the zipped folder title `Proj2190101_X_Y` and upload the single zipped folder to myCourseVille for your group before the deadline.

7. Grading criteria

In particular, your group work will be evaluated on the following metrics:

File	Class/Method	Criterion	Points of 100)
`CountryInfected.java`	`CountryInfected`	● Correct class definition	5

`Proj2190101_X_Y.java` `X_Y.csv`	`getInfected()`	● has the correct method header mentioned in Section 3 and works correctly on an on-going basis with the csv file being updated every day. ● must perform all data preprocessing as mentioned in Section 3.1 correctly and produce the correct output as mentioned in Section 3.	5 10
	`getSCurve()`	● has the correct method header mentioned in Section 4 and works for general input arguments: any time series, any bounds, and the number of days in the future. ● produce a feasible projection from the S-curve model. ● To get a full credit for time efficiency, it must not take longer than 10 minutes to finish. ● *Your projection for the 3 assigned countries in the csv file will be BASED ON THE MSE of your projected data and real data WHEN COMPARED TO THOSE OF THE OTHER GROUPS.*	5 10 5 15
	`getDoNothingCurve()`	● has the correct method header mentioned in Section 4 and works for general input arguments: any time series, and the number of days in the future. ● produce a correct projection from the Do-nothing model	5 15

	`main()`	- produce the correct csv file as mentioned in Section 5	10
`SCurve_X_Y.PDF`		- The report must not be longer than a page using standard 14-pt Thai Sarabun PSK font and margin. - produce a feasible explanation of how you fit(test) your S-curve model as mentioned in Section 4, 5	5 10

8. Questions

Although we are not supposed to give you any further help, as this final project is equivalent to the final exam. In case you do get really stuck and need some guidance. Please write to all of us simultaneously at sukree@gmail.com, duangdao.w@chula.ac.th, sirin.n@chula.ac.th, and cc all the group members, with the title: "Question Regarding 2190101-Computer Programming-Final Project-Sec X Group Y," where X and Y are your section and group number. State your question in the email concisely and directly. *Each group could ask only one question. We have the rights not to answer your second question or any question, should we think that the information in the project sheet is already sufficient. Also, we also have the rights not to answer, if you do not include all the emails (both recipients and cc) above in a single email or use the wrong email title.*

9. Plagiarism

This final project is to replace the final exam, so you CANNOT collaborate in any manner across groups. *Should we detect any code copy for any part of your code, we have the right to give you ZERO credit for that part or an F, and possibly escalate it to the university administration. Apart from that, since your group codes have to compete with those of the others, letting the other groups copy your codes/ideas would only mean that those groups could build on yours, resulting in your getting a lower grade.*