# PROJECT DESCRIPTION

## Description

In this project, students work in their project groups to solve an algorithmic problem from a given dataset involving:

- Using appropriate data structures/algorithm to solve a problem,
- running experiments and discussing the experimental outcomes
- analyzing the trade-offs of time and space efficiencies
- presenting the findings as a video presentation

## Timeline

- Week 9 Wed 12 Oct : Release of project requirements
- Week 12 Wed 2 Nov, 12pm noon: Project report via 10-min video, source codes + output file and youtube link

## Problem

In this course, we have looked at using hashcodes and hashmap for retrieving a specific (key, value) pair. The main objective is to minimize collisions (via generating a different hashcode where possible) so that we can quickly locate an object (value) within the hashmap based on the hashcode (key).

Interestingly, we can adopt the hashmap for a different problem, i.e., for retrieving similar values. In this case, we would make use of the hashmap such that similar objects are hashed into the same buckets instead. For e.g., suppose we are dealing with 16-character DNA sequences, where each character could only be A, C, T, or G. Given a DNA sequence "ACGTGTAC" and an error threshold of at most 2 characters, we would consider another DNA sequence "ACCTGTAT" similar. We would ideally generate the same hashcode instead for these 2 sequences considering their similarity and place them in the same bucket within the hashmap. If we have used the typical way of using hashmap, these 2 files would have generated a different hashcode and thus be placed at different buckets in the hashmap. The overall objective is thus to maximize collisions among similar objects and simultaneously to minimize colliisions among dissimilar objects.

## Source Codes

1. In the provided source codes, a simplistic implementation was given as a sample. In this example, a DNA sequence that starts with the same character are deemed to be similar and thus generated the same hashCode. Do understand the codes provided. The main method is in App.java.

2. Understand the implementations for App.java, DNASequence.java, TestEntry.java, ChainHashMap.java. The other classes are provided for reference.

3. You can only modify DNASequence.java. Do not modify any other .java files or create your own .java files to ensure that we would be able to run your codes after your submission. You will only submit DNASequence.java

4. data10000.txt contains 10000 unique strings of 16-character DNA sequence (e.g., CTAGATGTTTAGATGG). Do note that only 4 possible characters (A, C, T, G) are allowed in a sequence. You can assume it would always be 16 characters of DNA sequences in all test cases.

5. verify1000.txt contains 1000 unique strings of 16-character DNA sequence which would be used to test against your implementation for DNASequence.java. Every entry in this file would be used to generate a hashcode based on your implementation in DNASequence.java and would be used to compare for similarity with the other DNA sequences that was generated in the HashMap based on data10000.txt.

6. There's an error threshold, currently set as 6 in App.java. This is used to determine how many mismatched characters are allowed for 2 sequences to be deemed similar. This can be changed in the eventual test cases.

7. Amend the implementation of the hashCode() method in DNASequence.java. You are free to add additional private methods and attributes in InputFile.java. Do not modify the other methods and attributes in the DNASequence.java.

8. The following is a sample output based on a smaller dataset size (use data10.txt and verify5.txt) to help you understand how the summary of results are computed.

```
CCACAACCAAACATGG -> Similar : 1, Total(in same Bucket) : 3 = Similarity Score : 0.3333333333333333
GCAGTAGAATACAAGG -> Similar : 1, Total(in same Bucket) : 2 = Similarity Score : 0.5
AATCCGGTTTAGCTTG -> Similar : 1, Total(in same Bucket) : 4 = Similarity Score : 0.25
TTACCGAGGTAAACGA -> Similar : 1, Total(in same Bucket) : 1 = Similarity Score : 1.0
TAGCTCCCAACAAGCT -> Similar : 0, Total(in same Bucket) : 1 = Similarity Score : 0.0

----------------------------------------------------------------------
Summary of Results for data10.txt and verify5.txt
----------------------------------------------------------------------
Total Number of Verify Sequences : 5
Error Threshold : 6
Total Similarity Score : 2.08
Quality Indicator (Max 1.00) : 0.42
Total Number of Similar : 4
Total Number of Sequences (in same Bucket) : 11

Successfully written to output_data10.txt_verify5.txt_errorThreshold_6.txt
```

9. Your code submission will be tested against other test cases other than the ones that were provided.
10. You can only use the standard Java library classes.
11. Provide meaningful comments in your codes.
12. Comment out the print statements (if any) that you have used for debugging purposes.
13. Do look out for announcements in Piazza for any clarifications on requirements.
14. Ensure App.java can be executed within 2 minutes.
15. Please ensure that all your codes can be compiled with the given App.java file. Do not modify App.java file and all other .java files. You can only modify DNASequence.java. Non-conformance will lead to a penalty.

## Grading Criteria (project will account for 20% of the overall grade for the course)

- **Test Cases Quality Score (35%) :** results of algorithm against **other sets of data.txt, verify.txt and error thresholds,** based on:
  - Quality indicator
  - Number of valid similar sequences (to be maximized)
  - Number of sequences in the same bucket (to be minimized)
- **Video Presentation (50%) :** algorithm, experimentation, analyses, clarity of presentation

  We may be requesting some groups to discuss their projects in week 13 based on the submitted performance of results based on data10000.txt and verfiy1000.txt. Do note that the eventual grading would still be based on how the implementation works with other sets of test data.
- **X-factor (15%):** What makes the project outstanding. You are welcome to justify for your X-factors in the final video.
- **Note that even though this is a group project, we will be conducting peer evaluation to get feedback from the respective group members that will influence the class participation components.**

## Deliverables

Upload all deliverables by **Week 12 Wed 2 Nov, 12pm noon**

**Ms Teams**

(A) Project Video (.mp4 or .avi. or .mov format) (**Maximum 250mb**).

**Project report via video (max 10 min)**. Begin with the group name and members. Describe the problem. Describe the gists of the algorithm implemented and the key analysis of experimental results. Discuss the learning highlights, justify the x factors (what makes it special) and close with potential extensions. **Please record your video in normal playback speed.**

Name the video file as GXTX_Report.mp4 e.g., G1T1_Report.mp4

Create a video directory in your Group channel on MS Teams under Files and upload the video in the directory e.g. Files -> video/G1T1.mp4

Please housekeep your MS Teams channel and ensure that the total file size of your channel is less than 1GB.

Other than Ms Teams submission, upload the video to YouTube as an unlisted link, and provide the link via https://smu.sg/AY22CS201

**eLearn**

(A) Project Source Codes and Output File

(i) Submit DNASequence.java for the source code

(ii) Submit the generated file - output_data10000.txt_verify1000.txt_errorThreshold_6.txt

Multiple submissions are allowed but only the most recent submission will be kept in the system as the final submission.