Summarised Project Brief

💡 **Deadline: 2 Oct 2022, 5:00PM [Sunday]**

Requirements

Create **DBMS** with the following 2 requirements:

1. Storage:
    a. Part of main memory is allocated to be used as disk storage
    b. Disk capacity = **100 - 500MB**
    c. Disk access is in **blocks** as a unit
    d. Blocksize = **200B**
2. Indexing:
    a. B+ tree is used (*nodes are in memory bounded by blocksize*)

Implementation

- C/C++ is recommended but any programming language is fine.
- Experiments:
    - Store data and report:
        - number of blocks
        - size of database (*MB*)
    - Build B+ tree on attribute **numVotes** by inserting records sequentially and report the following statistics:
        - param **n** of B+ tree
        - **num of nodes** in tree
        - **height** of tree (*num of levels*)
        - **Content** of **root and 1st child** node of it.

- Retrieve movies of **numVotes = 500** and report:
    - **Num** and **content** of **index nodes** the process accesses
        - **Content** → first 5 index nodes/data if there are > 5
    - **Num** and **content** of **data blocks**
    - **averageRatings** of records.
- Retrieve movies with 30, 000 ≥ numVotes ≥ 40, 000 and report:
    - **num** and **content** of **index nodes** and **data blocks**
    - **averageRatings** of records.
- Delete movies with **numVotes = 1000**, update B+ tree and report:
    - the **number of times** that a **node is deleted** (or two nodes are merged) during the process of the updating the B+ tree;
    - the **number nodes** of the updated B+ tree;
    - the **height** of the updated B+ tree;
    - the **content of the root node** and its **1st child node** of the updated B+
- Re-set the block size to be **500 B** and **re-do Experiment 1, 2, 3, 4, and 5.**

Submission

1. **Report**
    a. It is suggested to use some figures to illustrate the designs and include the size information of fields and records.
    b. Design of the storage component, including:
        i. *how each data item is stored as a field,*
        ii. *how fields are packed into a record,*
        iii. *and how records are packed into a block.*
    c. Design of the B+ tree component, including the data structure of a node and the maximum number of keys a node maintains.
    d. Results of the experiments in Part 2;
    e. The contribution of each group member (presented at the first page of the report); and

> 💡 **Source code** (You must attach an installation guide to ensure that your code
> can be run successfully. **You will not receive any credit if your code fails to
> execute.**)

2. **Data**
    a. The data contains the IMDb rating and votes information for movies
        i. tconst (string) - alphanumeric unique identifier of the title
        ii. averageRating – weighted average of all the individual user ratings
        iii. numVotes - number of votes the title has received

    The first line in each file contains headers that describe what is in each column.

> 💡 *The data could be downloaded via this link: https://www.dropbox.com/s/c04kfatnd9lrtx9/data.tsv?dl=0*

3. **Submission policy.**
    a. All submissions should be uploaded to NTULearn (*a submission slot shall be created later on*).
    b. Late submissions will be penalised by **5% deduction per day** for at most **7 days**. Beyond 7 days after the deadline, no submissions will be accepted.

> ***It is not allowed to copy or refer to public code repositories. Strict plagiarism will be conducted. Any found plagiarism will mean a failing grade and be subject to further disciplinary actions. Some groups may be asked to demonstrate/explain their codes.***