# Homework #4 (Indexing and Query Execution)

## Due: 11:59pm, Friday, November 10, 2023

## Points: 100

1. [40 points] Consider storing a table person (id int primary key, name varchar(20), gender char(1)) in a CSV file named person.csv, where each line of the file represents a person record.
   1. Write a program insert.py that inserts a new person record into the table. After executing the program, the person.csv file should contain the new record. The program should not allow insertion to proceed if it violates the primary key constraints.

Execution format:

python3 insert.py <id> <name> <gender>

where <id> <name> <gender> represent specific values for the attributes. You should assume that person.csv (which might be empty) already exists before executing the command.

* 1. Write a program search.py that returns records of students with a given name.

python3 search.py <name>

1. [60 points] Consider natural-joining tables R(a, b) and S(a,c). Suppose we have the following scenario.
   * 1. R is a clustered relation with 10,000 blocks.
     2. S is a clustered relation with 20,000 blocks.
     3. 102 pages available in main memory for the join.
     4. Assume the output of join is given to the next operator in the query execution plan (instead of writing to the disk) and thus the cost of writing the output is ignored.

Describe the steps for each of the following join algorithms. For sorting and hashing-based algorithms, also indicate the sizes of output from each step. What is the total number of block I/O’s needed for each algorithm? Which algorithm is most efficient in terms of block’s I/O?

* 1. [10 points] (Block-based) nested-loop join with R as the outer relation.
  2. [10 points] (Block-based) nested-loop join with S as the outer relation.
  3. [20 points] Sort-merge join (assume only 100 pages are used for sorting and 101 pages for merging). Note that if join cannot be done by using only a single merging pass, runs from one or both relations need to be further merged, in order to reduce the number of runs. Select the relation with a larger number of runs for further merging first if both have too many runs.
  4. [20 points] Partitioned-hash join (assume 101 pages used in partitioning of relations and no hash table is used to lookup in joining tuples).Note if buckets are still too large to join within memory, you should further partition them.

**Permitted Library:** sys

**Submission Instructions:**

* 3 files: insert.py, search.py, q2.pdf
* DO NOT submit zip file.
* Highlight the final answers for each part (a,b,c,b) in q2.pdf.
* Each part of question 2 has only 1 approach to answer. Multiple approaches will not be accepted in answer.
* All assumptions are clear in questions. No extra assumptions should be made.
* If queries/code did not run for q1, then 0 points.