CS 409 - UG SOFTWARE LAB
Assignment 3 Weightage: 25%
Submission Deadline: 11-Mar-2022 11:59pm

General Instructions:

- You must use only C, C++, Python or JAVA for this assignment.
- Prescribed specifications must be strictly followed. Failure to do so may lead to substantial loss of points.
- Make sure your code is well written (self explanatory variable names) and documented. You are likely to lose points if your TA cannot understand your code.
- You must submit a README file along with your code with clear instructions on how to run the code.
- Any assumptions made must be clearly stated in a README file. Suitable ones would be accepted and graded accordingly.
- **You are not allowed to use any specialized libraries in this assignment. You are allowed to use only the primitive data types in the language such as arrays, structures, classes, hash_map, associative array, etc. You may use libraries for implementing priority queues, min heaps, etc.**
- **Your code must not have any directory structure in it. All the code files must be inside a single directory. Also, you must include a readme file on how to run your code.**

Question 1 (100 points) Secondary Memory Based Implementation of Shortest Path algorithms

For this assignment you would be using the partitions developed in Assignment 2. Develop a shortest path algorithm implementation (Dijkstra's) which runs over the partitions developed previously. Input to the algorithm consists of a source node id and destination node id.

Managing Page Faults: As your Dijsktra's algorithm proceeds, it would be reading "disk blocks' ' (simulated as files in your code) corresponding to various cell-ids (refer to lecture videos). Ideally, such a system should have a main memory buffer manager to manage the "pages/disk blocks" read into the main memory. However, in this assignment, we would assume that we have unlimited space in the main memory for bringing in the "disk blocks." Nevertheless, for every request for accessing the portion of the graph corresponding to a cell, your code must first check if it's information is already present. In case, it's not present then its "disk block(s)" should be read from the files.

Hint: Maintain the "graph-seen" so far in the form of an adjacency list. Whenever disk blocks corresponding to a cell are brought in, they should be used to update the "graph-seen" data structure. Also, your code must maintain the list of cell-ids which have been brought into main memory. This way, whenever your code requests for a cell-id, you can easily determine if its information is already present in the graph constructed so far.

For every shortest path query, your code must display the final shortest path, its length and the number of disk blocks brought into the main memory.

Evaluation: TAs would evaluate your code on both small and large datasets. As always, evaluation on small datasets would be to ensure correctness. Whereas, large datasets would be used to test scalability. You are also required to implement normal Dijsktra's algorithm which does not use the previously mentioned partitions. We would be using the output of this to check the correctness of your code. In other words, the shortest path returned by Dijsktra's (for any particular query) implemented on secondary memory should be of the same length as that of normal Dijsktra's algorithm.