Aligned subject learning outcomes	· **SLO1 Evaluate the efficiency and effectiveness of data structures and algorithms** · **SLO3 Assess and apply suitable data structures and algorithms to IT systems and applications** · **SLO4 Design and implement data structures and algorithms**
Group or individual	*Individual*
Weighting	*30%*
Due date	Week 12, Friday 5pm

Databases are IT systems that require high-performance data structures to provide efficient searching and updating operations. Databases often use balanced trees or hash tables to provide **indexing** so that rows can be very quickly located. In this assignment, you will evaluate the efficiency of some of the data structures we have covered to make judgements about their suitability for their use within a database application.

Use sample data to build maps based on:

- A probing hash table (ch10/probe_hash_map.py)
- An AVL tree (ch11/avl_tree.py)
- A modified probing hash table (that you are to implement) that stores the hash value as an extra field.

Base your modified probing hash table on exercise C-10.34:

> Computing a hash code can be expensive, especially for lengthy keys. In our hash table implementations, we compute the hash code when first inserting an item, and recompute each item's hash code each time we resize our table. Python's dict class makes an interesting trade-off. The hash code is computed once, when an item is inserted, and the hash code is stored as an extra field of the item composite, so that it need not be recomputed. Reimplement our HashTableBase class to use such an approach.

Perform experiments to determine the running time necessary to build these structures (perform multiple runs to determine an average), and then the running time needed to find values given a randomly selected key. Again, perform multiple runs to determine an average.

Provide a discussion of your results. Based on your findings, which implementation do you think is preferable under different circumstances, such as frequent updates vs. infrequent updates? Is modifying the probing hash table a worthwhile change?

After you have developed your own ideas based on your experiments, research the methods that real-world databases use to create indexes and store their data. Relate your experimental results to your research findings. Do your results match what you have found in your research? Consider questions such as:

- When are ordered maps more useful vs. unordered maps?
- What additional optimisations can you make to a hash table implementation to improve its performance?

Deliverables

1. **Python code for your modified probing hash table and experiments. Include all Python files in a single ZIP folder so they can be easily run for marking.**
2. **A Word document including your results, discussion, and research findings. Your results and discussion should be about 3 pages or 1500 words.**

Rubric

Criteria	**Excellent**	**Good**	**Marginal**	**Unacceptable**
Implementation of modified ProbeHashMap	**10** Correct implementation of modified ProbeHashMap to effectively store hashed keys and avoids recalculation when resizing.	**7.5** Implementation of modified ProbeHashMap effectively stores hashed keys.	**5** Imlementation of modified ProbeHashMap stores hashed keys..	**0** Modified ProbeHashMap does not store hashed keys.
Implementation of experiments	**20** Python code for experiments is effective and well-documented.	**15** Python code for experiments is generally effective and well-documented.	**10** Python code for experiments is effective in some cases and some documentation is provided.	**0** Python code for experiments does not work and lacks any documentation.
Experimental results and discussion	**20** Experimental results are thorough and presented in a clear format. Discussion effectively links the experimental results to knowledge of the data structures and their efficiency.	**15** Experimental results are presented in a clear format. Discussion links the experimental results to knowledge of the data structures and their efficiency.	**10** Experimental results are presented. Discussion relates results to some knowledge of data structures.	**0** Experimental results are lacking or unclear. Discussion fails to relate results and knowledge of data structures.
Research into real-world databases	**20** Clear and detailed explanation of real-world issues involved in implementing databases and the implications on what data structures and algorithms are used. References are provided in a clear and consistent format.	**15** Generally clear explanation of real-world issues involved in implementing databases and the implications on what data structures and algorithms are used. References are provided in a consistent format.	**10** Explanation of real-world issues involved in implementing databases. References are provided in a consistent format.	**0** Explanation lacks substance or is incomprehensible.