**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | Algorithms and constructs |
| **Assessment Title:** | CA2 |
| **Lecturer Name:** | Dr. Muhammad Iqbal |
| **Student Full Name:** | Bolor-Erdene Sainbileg |
| **Student Number:** | 2024648 |
| **Assessment Due Date:** | Monday 7th Apr 2025 |
| **Date of Submission:** | 10th May 2025 |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

I decided to use recursive insertion sort to sort employee records and a binary search employee by name efficiently. The choices were influenced by the structure of application, ease of implementation, and the size of the dataset we were about to handle.

Insertion sort was used because it is efficient for small to medium-sized datasets, which is typical for most organizations. The number of employees in a company is typically several dozen too few hundreds, not millions. Insertion sort especially effective when the data is already nearly sorted, which is typically the case with employee records that are updated or maintained on a regular basis. Instead of re-ordering all of them, insertion sort is the kind where new records can be added and repositioned to the correct position without a lot of shifting. I employed recursive implementation since it makes the logic cleaner and easier to comprehend. Such cleaner alignment is especially helpful within an academic environment, where it is as important to comprehend clearly the logic behind the algorithm as it is to be efficient.

For searching, we employed binary search since it is both fast and efficient. Binary search is effient with ordered data and has a substantial performance improvement over linear search, particularly when the number of records is high. Rather than testing each item individually, it keeps in dividing the list of halves, thus minimizing the number of comparisons significantly. Even though my data is not large now, this method keeps the application responsive as more records are inserted over time. It also satisfies the user’s expectation – user demand immediate search results, and binary search allows us to deliver that smooth experience.

These two algorithms together provide you with a good basis for keeping employee records. They are both basic algorithms, being efficient and easy to use. Most importantly, they are easy to implement and maintain, which is critical for project like this. By emphasizing simplicity, readability and good performance, we ensured that our solution would be enough for the project needs without complicating the code too much.

Briefly, recursive insertion sort and binary search were ideal choices for my assignment. They introduced a real-life, efficient and readable solution that facilitates the functionality as well as maintainability of the code.

Github Link:

[**https://github.com/Bolor-e/CA2.git**](https://github.com/Bolor-e/CA2.git)