|  |  |  |  |
| --- | --- | --- | --- |
|  | FPT UNIVERSITY IN HOA LAC  Class: CS1401  Major: Computer Science  Course: CSD201 | Káº¿t quáº£ hÃ¬nh áº£nh cho FPT university logo |  |
| assignment 1  group: groupname | | | | |
| Project Summary | | | | |
|  | | | | |

|  |  |  |
| --- | --- | --- |
| Report Date | aSSIGNMENT NAME | Prepared By |
| September 26th , 2019 | **Assignment 1.3:** Tax Management | **Hà Long Duy**: [duyhlhe141012@fpt.edu.vn](mailto:duyhlhe141012@fpt.edu.vn)  **Lê Đức Anh**: [anhldhe140935@fpt.edu.vn](mailto:anhldhe140935@fpt.edu.vn)  **Lê Xuân Huy**: [huylxhe140555@fpt.edu.vn](mailto:huylxhe140555@fpt.edu.vn) |

|  |
| --- |
| ASSIGNMENT TASK SUMMARISON |
|  |

**Create a console program to manage tax payers’ information**

|  |
| --- |
| TASK MANAGEMENT |
|  |

|  |  |  |
| --- | --- | --- |
| member | Main task | sOLUTION (data structures & algorithms) |
| **Hà Long Duy** | File-related Methods (Load & save file) | Linked List  Bubble Sort  Linear Search |
| **Lê Đức Anh** | Main coder |
| **Lê Xuân Huy** | Report writer + tester |

|  |
| --- |
| Implementation |
|  |

|  |  |
| --- | --- |
| IDE | GITHUB REPOSITORY |
| **Netbeans IDE 8.2** | <https://github.com/leeanh38/CSD201/tree/master/Tax.Management> |

|  |
| --- |
| EVALUATION (complexity analysis) |
|  |

|  |  |  |
| --- | --- | --- |
| CONTENT | EFFectiveness | |
| Time complexity | comparison |
| **Bubble Sort** | **Worst:** O(n2) **; Best:** O(n) | **Mergesort:** is preferred for sorting linked list   * **Time complexity for all case:** O(nlog(n)) * **Space complexity:** O(n)   **Quicksort:** perform poorly due to slow random-access performance of linked list   * **Worst:** O(n2) **; Other cases:** O(nlog(n)) * **Space complexity:** O(log(n))   **Other sorting algorithms:** impossible to be used in linked-list sorting |
| **Linear Search** (perform equality comparisons) | O(n) | **Binary search:** O(log(n)) **-** perform ordering comparisons |
| **Singly-linked List** | **Methods: (Worst case)**   * isEmpty(): O(1) * length(): O(n) * **addToEnd():** O(1) * display(): O(n) * **search():** O(n) * checkExistCode() : O(n) * **delete():** O(n) * addAfterPosition(): O(n) * deleteByPosition(): O(n)   **Space complexity:** O(n) *(applied to every types of list)* | **Array:** have to declare size before being implemented, problematic when it comes to inserting and deleting middle element (move elements around)   * **search():** O(n) * **insertion():** O(n) * **deletion():** O(n)   **Stack & Queue:** LIFO & FIFO data structure. Not suitable for random-access performance   * **search():** O(n) * **insertion():** O(1) * **deletion():** O(1) |