Task 1 Report – Patient List (Linked List)

In this task, a patient management list is implemented by using a singly linked list data structure. The main purpose of this task is to understand how linked lists work and how basic operations are done on them. Each patient is stored as a node in the list and contains patient information such as id, name, severity and age.

The linked list supports four main operations: addPatient, removePatient, findPatient and printList. The addPatient method adds a new patient to the end of the list. This operation has **O(n)** time complexity because in worst case the program needs to go through all nodes to reach the last one. The removePatient method also has **O(n)** complexity, since the list must be traversed to find the patient with the given id. Similarly, the findPatient method is **O(n)** because it searches each node one by one until the patient is found or the list ends.

The printList operation also runs in **O(n)** time because it prints all patients in the list. In general, all basic operations in this linked list require linear time.

When comparing **Linked List** and **ArrayList**, both have advantages and disadvantages. Linked Lists are better when frequent insertions and deletions are needed, because adding or removing elements does not require shifting other elements. However, searching for an element is slower because there is no direct access by index. On the other hand, ArrayList allows fast access with indexes (**O(1)**), but insertion and deletion can be costly due to shifting elements.

In this task, Linked List is a suitable choice because patient records can be added or removed dynamically and the list size can change easily.