**Exercise 6: Library Management System**

**1) Explain linear search and binary search algorithms.**

**Linear Search:** Linear search is a straightforward algorithm used to find a specific element in a list. It works by checking each element in the list one by one from the beginning to the end until the desired element is found or the list is fully traversed. If the element is found, the algorithm returns its position; if not, it returns a signal indicating that the element is not in the list. This method is simple to implement and works on both sorted and unsorted lists but can be inefficient for large datasets due to its O(n) time complexity, where n is the number of elements in the list.

**Binary Search:** Binary search is a more efficient algorithm used for finding a specific element in a sorted list. It works by repeatedly dividing the search interval in half. Initially, the search interval covers the entire list. The algorithm compares the target value to the middle element of the list. If they are equal, the search is complete. If the target value is smaller, the search continues in the left half; if larger, it continues in the right half. This process repeats until the target value is found or the interval is empty. Binary search has a time complexity of O(log n), making it much faster than linear search for large datasets, but it requires the list to be sorted beforehand.

**2) Compare the time complexity of linear and binary search.**

**Linear Search:** The time complexity of linear search is O(n), where n is the number of elements in the list. This means that in the worst case, the algorithm might need to check every single element in the list once to find the target or determine that it is not present. As a result, the time taken grows linearly with the size of the dataset, making it less efficient for large datasets.

**Binary Search:** The time complexity of binary search is O(log n), where n is the number of elements in the list. This means that the algorithm reduces the search interval by half each time it checks the middle element. Consequently, the time taken grows logarithmically with the size of the dataset, making binary search significantly faster than linear search for large datasets. However, binary search requires the list to be sorted beforehand, which might add additional preprocessing time if the list is initially unsorted.

**3) Discuss when to use each algorithm based on the data set size and order.**

**Linear Search:** Use linear search when dealing with small or unsorted datasets. It checks each element one by one, so it doesn't require the data to be in any specific order. For smaller datasets, the performance difference between linear and binary search is negligible, and linear search is simpler to implement. It's also useful when the data set is frequently updated, as sorting the data for binary search would be inefficient.

**Binary Search:** Binary search is ideal for large, sorted datasets. It works by repeatedly dividing the search interval in half, which makes it much faster than linear search for large data sets. However, it requires the data to be sorted beforehand. If the data is not already sorted, sorting it first can be time-consuming, so binary search is most effective when the data set is static or changes infrequently.