TASK

Sorting and Searching

Model Answer Approach

Visit our website

Auto-graded Task 1

In this approach, we have a class named Album which represents a single album that is used to create instances inside the **album1** and **album2** collections. The approach is well-suited for handling the albums and their attributes in a structured manner. It utilises user-defined functions to perform the required operations as well as to enhance the code organisation and reusability.

Auto-graded Task 2

The program implements a modified merge sort algorithm to sort lists of strings in descending order based on their lengths. It defines a function **merge_sort_by_length()** that recursively splits the list into halves, sorts each half, and then merges them based on the length of the strings. Additionally, there's a helper function **merge_by_length()** to merge the sorted halves. Some students might overlook the importance of handling the base case, which can lead to an infinite recursive call.

Auto-graded Task 3

This approach defines a function called **linear_search()** to search for the target number in the given unsorted list. It then utilises an **insertion_sort()** function to sort the list of integers in ascending order so that the **binary_search()** function can be performed. A common mistake students might make is forgetting that a binary search, although quicker, requires a sorted list.