TASK SHEET 4

1. Develop a function that efficiently finds the longest common subsequence between two strings.
2. Write a program that efficiently computes the nth Fibonacci number using memoization.
3. Implement a function that generates all possible permutations of a list.
4. Create a program that efficiently checks if a given string is a palindrome.
5. Design a function that efficiently calculates the determinant of a square matrix.
6. Develop a program that efficiently sorts a list of integers using the radix sort algorithm.
7. Write a function that efficiently solves the Tower of Hanoi puzzle for n disks.
8. Implement an algorithm to find the shortest path in a weighted graph using Dijkstra's algorithm.
9. Create a program that efficiently checks if a given number is prime.
10. Design a function that efficiently calculates the Levenshtein distance between two strings.
11. Develop a program that efficiently finds the median of a list of integers.
12. Write a function that efficiently checks if a given graph is acyclic (contains no cycles).
13. Implement an algorithm to efficiently find all possible combinations of a given list of integers that sum up to a target value.
14. Create a program that efficiently solves the subset sum problem using dynamic programming.
15. Design a function that efficiently determines if a given binary tree is a binary search tree.