**Task1:**

This code implements Kruskal's algorithm to find the minimum spanning tree of a given graph and then writes the calculated minimum cost to an output file.Kruskal's algorithm greedily selects edges of the graph while ensuring that no cycles are formed, ultimately leading to a minimum spanning tree with the lowest possible total weight.The code sorts the edges in the graph list based on their weights using the sort method with a key function. This sorting is essential for Kruskal's algorithm, as it ensures that edges are considered in ascending order of weights, facilitating the construction of a minimum spanning tree. The lambda function lambda x: x[2] extracts the weight from each edge tuple, and the sorting operation arranges the edges accordingly.

**Task2:**

This code reads an input value 'n' from 'input2.txt', calculates the 'n'-th term of a sequence using a Fibonacci-like recurrence relation, and writes the result to 'output2.txt'. The function fibo(n) calculates the value of the 'n'-th term by building a dynamic programming array step, where each term is the sum of the previous two terms. The final result is the 'n'-th term of this sequence. The calculated result is then written to the output file.

**Task3:**

This code reads input values from 'input3.txt', representing the number of coin denominations n and the target value x, as well as a list of coin denominations. It then calculates the minimum number of coins required to achieve the target value x using the provided coin denominations. The min\_coin function employs dynamic programming to iteratively determine the minimum number of coins needed for each value from 1 to x. The final result is written to 'output3.txt'. This code effectively solves the coin change problem, finding the smallest number of coins needed to make up the target value x using the given coin denominations.