Explanation of the Code

Overview

- The code calculates the number of different assignments that can be made between **n** students and **n** tasks, given the preferences of each student for each task.
- The code uses a permutation-based approach to generate all possible assignments and checks each assignment for validity based on the student preferences.

Functions

- **swap Function**
    - Purpose: Swaps the values of two integers using a temporary variable.
    - Parameters: Two integer pointers **a** and **b**.
    - Implementation: Temporarily stores the value of a in a variable, assigns the value of b to a, and then assigns the temporary value to b.
- **next_permutation Function**
    - Purpose: Generates the next permutation of an array of integers in lexicographic order.
    - Parameters: An integer array arr and its size n.
    - Implementation:
    - Finds the largest index i such that arr[i - 1] < arr[i].
    - If no such index is found, returns 0 to indicate that the last permutation has been reached.
    - Finds the largest index j such that arr[j] > arr[i - 1].
    - Swaps the values at indices i - 1 and j.
    - Reverses the suffix starting at index i to generate the next permutation.
    - Returns: 1 if a new permutation is generated, 0 if there are no more permutations.
- **calculate_assignments Function**
    - Purpose: Calculates the number of different assignments that can be made between n students and n tasks, given the preferences of each student for each task.
    - Parameters: An integer n representing the number of students and tasks, and a 2D integer array preferences representing the student preferences.
    - Implementation:
    - Initializes a temporary array temp with values from 0 to n - 1.
    -
    -
    - Uses a do-while loop to generate all possible permutations of the tasks using the next_permutation function.
    - For each permutation, checks if the assignment is valid by iterating through the student preferences and checking if each student is assigned a task they like.
    - If the assignment is valid, increments a counter count.
    - Returns: The total number of valid assignments.
- **main Function**
    - Purpose: Reads the number of students and their preferences from the user, calls the calculate_assignments function, and prints the result to the user.

- Implementation:
- Reads the number of students **n** from the user.
- Initializes a 2D integer array **preferences** to store the student preferences.
- Reads the preferences of each student from the user and stores them in the **preferences** array.
- Calls the **calculate_assignments** function with the **n** and **preferences** array as arguments.
- Prints the result to the user.

- **Key Points**
  - The code uses a permutation-based approach to generate all possible assignments, which ensures that all possible combinations of tasks are considered.
  - The next_permutation function is used to generate all possible permutations of the tasks in lexicographic order.
  - The calculate_assignments function checks each permutation for validity based on the student preferences, which ensures that only valid assignments are counted.
  - The code uses a temporary array to store the permutations of the tasks, which allows for efficient generation and checking of permutations.
  - The code uses a do-while loop to generate all possible permutations of the tasks, which ensures that all possible combinations of tasks are considered.
  - The code uses a nested for loop to read the preferences of each student from the user, which allows for efficient input of the student preferences.
  - The code uses a nested for loop to check if each assignment is valid, which ensures that all possible assignments are checked for validity.