Assignment 2: Module 4 and 5 Report:

- This C code is designed to calculate the number of possible assignments of subjects to students based on their preferences using dynamic programming. Let's break down the code and its functionality step by step:
- MX is defined as 100, which represents the maximum size for arrays and masks used in the program.

  Variables:
- mx: Represents the maximum possible bit mask value (1 << n calculates 2^n).
- dp[mx]: An array used for dynamic programming where dp[mask] stores the number of ways to assign subjects when mask represents the current state of assignment.
- memset(dp, 0, sizeof(dp)): Initializes dp array with zeros.
- dp[mx - 1] = 1;: Base case initialization where all subjects are assigned.

- **Outer Loop** (for (int mask = mx - 1; mask >= 0; mask--)):
  - mask iterates through all possible bit masks representing different assignment states.
  - s counts the number of subjects assigned in the current mask.

- **Inner Loop** (Assigning subjects):
  - Checks each subject i against the current assignment state (mask).
  - preference[s][i] checks if subject i is preferred by s students.
  - !(mask & (1 << i)) ensures subject i is not already assigned in mask.
  - If both conditions are satisfied, dp[mask] is updated based on possible assignments including subject i.

- **Return Value**:
  - dp[0] returns the number of ways to assign all subjects starting from no assignments (mask = 0).

Function: main

- **Variables**:
  - n: Number of students and subjects.
  - preference[MX][MX]: 2D array storing preferences where preference[i][j] is 1 if student i prefers subject j.
- **User Input**:

    - Asks for the number of students (n).
    - Reads preferences of each student for each subject into the preference array.
- **Output**:
    - Calls assign(n, preference) to compute the number of valid assignments.
    - Prints the result.

- **Dynamic Programming Approach**:
  The problem is solved using a dynamic programming approach where dp[mask] represents the number of ways to assign subjects represented by mask.
- mask is iterated from mx - 1 (all subjects assigned) to 0 (no subjects assigned).
- At each mask, the algorithm checks which subjects can be added next based on student preferences and updates dp[mask] accordingly.

  Bitmasking:
- Bitmasking is utilized to efficiently represent and manipulate subsets of subjects.
- mask | (1 << i) adds subject i to mask.
- (mask & (1 << i)) checks if subject i is already assigned in mask.
- **Complexity**:
- The algorithm runs in O(n * 2^n) time complexity due to iterating over all possible assignments (2^n states) and checking preferences for each state.

This program uses dynamic programming to efficiently compute the number of valid subject assignments based on student preferences. It handles input for the number of students, their preferences, and outputs the total number of valid assignments. The use of bitmasking (mask) and dynamic programming (dp) ensures that the solution is both efficient and concise for the given problem constraints.