assign function:

- This function calculates the number of different assignments of n different topics to n students such that each student gets exactly one topic they like.
- It uses dynamic programming to solve the problem.
- The function takes two parameters: `n` (the number of students) and `preference` (a 2D array of size n x n, where `preference[i][j]` is 1 if student i likes topic j, and 0 otherwise).
- The function initializes a dynamic programming table `dp` of size `2^n`, where `dp[i]` represents the number of ways to assign topics to students `s` to `n` given the state `i`.
- The function fills up the `dp` table in a bottom-up manner. For each state `mask`, it calculates the number of ways to assign topics to students `s` to `n` by considering each topic that the current student likes and has not been assigned yet.
- The function returns `dp[0]`, which represents the number of ways to assign topics to all students.

main function:

- The program reads the number of students `n` from the user.
- It initializes a 2D array `preference` of size n x n to store the preferences of each student.
- The program then reads the preferences of each student from the user and stores them in the `preference` array.
- The program calls the `assign` function to calculate the number of different assignments.
- Finally, the program prints the total number of assignments that can be prepared.

Dynamic Programming:

- The dynamic programming table `dp` is used to store the number of ways to assign topics to students `s` to `n` given the state `i`.
- The state `i` is represented as a bitmask, where the `j`-th bit of `i` is set if topic `j` is already assigned to a student.
- The function fills up the `dp` table in a bottom-up manner, starting from the state where all topics are assigned (i.e., `mask = 2^n - 1`).
- For each state `mask`, the function calculates the number of ways to assign topics to students `s` to `n` by considering each topic that the current student likes and has not been assigned yet.

- The function uses the previously calculated values in the `dp` table to calculate the new values.

Time and Space Complexity:

- The time complexity of this program is O(n*2^n) because it iterates over all possible states of the assignments.
- The space complexity is O(2^n) because it uses a dynamic programming table of size `2^n`.