ASSIGNMENT

LOGIC AND EXPLANATION OF THE CODE:

EASY 3:

Function Definition (generate_pascals_triangle):

- The function takes one argument, numRows, representing the number of rows to generate in Pascal's Triangle.
- If numRows is 0, it returns an empty list since Pascal's Triangle with 0 rows is an empty triangle.
- Initializes the result list with the first row, which is [1].

For Loop (for i in range(1, numRows):):

- Iterates from the second row up to the specified number of rows (numRows).
- Accesses the previous row in the result (prev_row).

Calculating the New Row (new_row):

- The new row starts with 1 (the first element of every row in Pascal's Triangle).
- The middle elements are calculated using a list comprehension:
    - prev_row[j - 1] + prev_row[j] for each j in the range from 1 to i.
- The new row ends with 1 (the last element of every row in Pascal's Triangle).

Appending the New Row to the Result (result.append(new_row)):

- The newly calculated row is appended to the result, effectively building Pascal's Triangle row by row.

Function Return (return result):

- The function returns the complete Pascal's Triangle as a list of lists.

User Input and Validation:

- The user is prompted to enter the number of rows they want in Pascal's Triangle.
- Checks if the input is within the valid range (1 to 30).

Generating and Printing Pascal's Triangle:

- If the input is valid, it calls the generate_pascals_triangle function and prints the resulting triangle.

Error Handling:

- If the input is not valid, it prints an error message.

MEDIUM 3:

Function Definition (maximalSquare):

- The function takes a 2D matrix (matrix) as input.
- Checks if the matrix is empty (i.e., no rows or no columns), and if so, returns 0.

Initialization:

- Obtains the dimensions of the matrix (m x n).
- Initializes a 2D array dp (dynamic programming table) with dimensions (m+1) x (n+1) and initializes all values to 0. This table is used to store the maximum side length of a square ending at the corresponding position in the original matrix.

Dynamic Programming Loop:

- Iterates through each cell in the matrix and updates the dynamic programming table (dp).
- If the current cell in the original matrix is '1':
    - dp[i][j] is set to the minimum of the values to the left, above, and diagonally above-left of the current cell in the dp table, plus 1.
    - Updates the max_side variable with the maximum value encountered so far.

Result Calculation and Return:

- The function returns the area of the largest square, which is calculated as max_side * max_side.

Example Usage:

- Three example matrices (matrix1, matrix2, and matrix3) are provided, and the maximalSquare function is called on each of them.
- The results (output1, output2, and output3) are printed.

HARD 2:

Function shortestPalindrome(s):

- The function takes a string s as input.
- It initializes two pointers, i and j. The variable i starts at the beginning of the string (0), and j starts at the end of the string (len(s) - 1).
- It iterates over the characters of the string from the end (j) towards the beginning (i), comparing characters at positions i and j.
    - If s[i] is equal to s[j], it means that the characters match. The i pointer is incremented (i += 1).
    - The loop continues until the characters at positions i and j no longer match or until the entire string is traversed.
- After the loop, the code checks if i has reached the end of the string (i == len(s)). If it has, the input string is already a palindrome, and the function returns the input string.
- If i is less than the length of the string, it means that there are characters remaining in the suffix that need to be added to the beginning of the string to make it a palindrome.
- The variable suffix is assigned the substring of s starting from index i.
- The function returns the concatenation of the reversed suffix, the result of a recursive call to shortestPalindrome with the substring s[:i], and the original suffix.

User Input and Validation:

- The script takes user input for a string (user_input).
- It checks if the input consists only of lowercase English letters using user_input.isalpha().
- If the input is valid, it calls the shortestPalindrome function with the user input and prints the result.
- If the input is invalid (contains non-alphabetic characters), it prints an error message.

Example Usage:

- The user is prompted to enter a string.
- The script then prints the result of the shortestPalindrome function applied to the user's input.