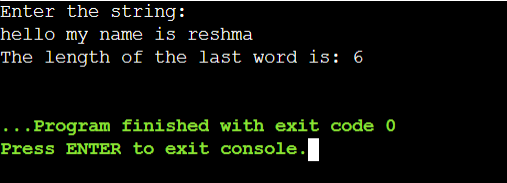
EASY - QUESTION 1:



LOGIC :

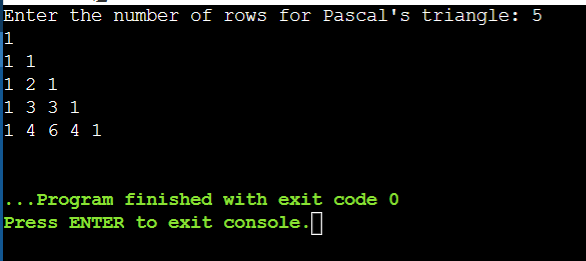
* Input Parameters: The function takes a character array (char[]) as its parameter. This array represents the input string for which we want to find the length of the last word.
* Initialize Variables: The function initializes two variables:

count: Represents the total length of the input string, obtained using the strlen function.

result: Represents the length of the last word, which will be the final output.

* Loop through the String: The function then enters a for loop that starts from the end of the string (count - 1) and iterates backward. The loop continues until it reaches the beginning of the string (index 0).
* Check for Non-Space Characters: Inside the loop, there is an if statement that checks whether the character at the current index (str[i - 1]) is not a space. If the character is not a space, it means it belongs to the last word, so the result variable (length of the last word) is incremented.
* Break at First Space: If a space is encountered, it indicates the end of the last word. In this case, the loop is terminated using the break statement.
* Return Result: Finally, the function returns the calculated length of the last word (result).
* Print Result in main Function :In the main function, the user is prompted to enter a string using fgets. The lastwordcount function is called to calculate the length of the last word, and the result is printed.

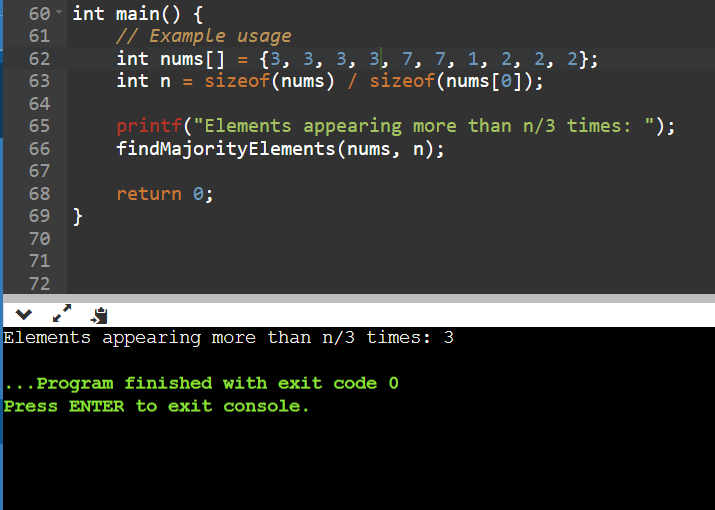
EASY - QUESTION 3:



LOGIC :

* binomialCoefficient Function: This function is responsible for calculating the binomial coefficient C(n, k) for a given pair of indices (n, k) using dynamic programming. It uses a 2D array C to store the intermediate results, where C[i][j] represents C(i, j).The nested loops iterate over all possible pairs (i, j) within the specified bounds, and the values are computed based on the recurrence relation for binomial coefficients : C[n,k] = C(n-1,k-1) + C(n-1,k).
* generatePascalsTriangle Function:This function takes the number of rows (numRows) as input and prints the corresponding rows of Pascal's Triangle.It uses a nested loop structure to iterate over each row and column, printing the binomial coefficient for each pair (i, j).
* main Function:The main function is the entry point of the program.It prompts the user to enter the number of rows they want in Pascal's Triangle using scanf.It then calls the generatePascalsTriangle function with the user-provided input to print the desired number of rows.
* Pascal's Triangle:Pascal's Triangle is a mathematical construct where each number in the triangle is the sum of the two directly above it. The first few rows of Pascal's Triangle represent binomial coefficients, and it has various applications in combinatorics.

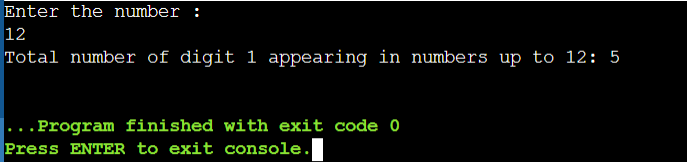
MEDIUM - QUESTION 2 :



LOGIC :

* Initialization: Initialize two candidate variables (candidate1 and candidate2) and their corresponding counters (count1 and count2) to keep track of the frequencies of the potential majority elements. Initially, both counters are set to 0.
* Step 1: Find Candidates: Iterate through the array to find potential candidates for majority elements. If the current element matches either candidate, increment the corresponding counter. If the counters are zero, update the candidate with the current element and set the counter to 1. If the counters are non-zero, decrement both counters.
* Step 2: Count Occurrences of Candidates: Reset counters to 0. Iterate through the array again to count the occurrences of the two potential majority elements.
* Step 3: Check if Candidates Appear More than n/3 Times: Check if the counts of candidates are greater than n/3. If yes, print the candidates as they are the elements appearing more than n/3 times in the array.
* Example Usage in main Function: In the main function, an example array (nums) is provided with some repeated elements. The findMajorityElements function is called with the array and its size to identify and print elements appearing more than n/3 times.
* Example Output: For the given example array {3, 3, 3, 3, 7, 7, 1, 2, 2, 2}, the output will be the elements that appear more than ⌊n/3⌋ times, which is 3 in this case.

HARD - QUESTION 3:



LOGIC :

* Input:The function takes an integer n as its parameter, representing the upper limit of the range of numbers to consider.
* Loop Over Numbers from 1 to n:The function uses a for loop to iterate over all numbers from 1 to n. The loop variable i represents the current number being processed.
* Count Digit "1" Occurrences in Each Number:Within the loop, the function enters another loop using a while loop.The inner loop processes each digit of the current number i by repeatedly dividing num by 10 until num becomes zero.For each digit, the function checks if it is equal to 1 (num % 10 == 1). If true, it increments the count variable, indicating that a digit "1" has been found in the current number.
* Return Total Count: After processing all numbers from 1 to n, the function returns the total count of occurrences of the digit "1" in this range.
* Example Usage in main Function: In the main function, the user is prompted to enter a number (n) for which the count of digit "1" occurrences will be calculated. The countDigitOne function is then called with this input, and the result is printed.