Recursion - Exercise

Write recursive functions in C to solve the following problems and analyse their time and space complexity.

Problem 1:

Given a number 'n', determine the number of digits in 'n'.

Function Prototype: int numDigits(int n);

Sample: input: 1234123, output: 7

Problem 2:

Given two numbers n and k, print n tables from 1 to k.

Function Prototype: void tables(int n, int k);

Sample: input: n = 5, k = 10,
output:
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25
6 * 5 = 30
7 * 5 = 35
8 * 5 = 40
9 * 5 = 45
10 * 5 = 50

Problem 3:

Given a string comprising of unique characters, print in sorted order all unique permutations of all characters in the string.

Function Prototype: void permutations(char *string);

Sample: input: ABC, output: ABC ACB BAC BCA CAB CBA

Problem 4:

Given a number 'n', compute the following:

$$\prod_{i=1}^{2} \sum_{j=\frac{(i-1)i}{2}+1}^{i(i+1)/2} j$$

Function Prototype: long int pdtOfSums(int n);

Sample: input: 4, output: 2550

Explanation: 1 * (2+3) * (4+5+6) * (7+8+9+10)

Problem 5:

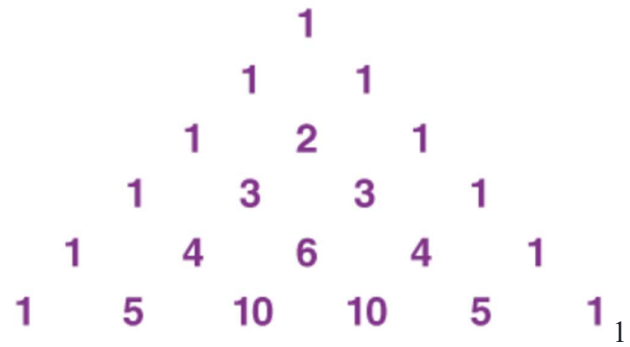
Given a number ‘n’, print the first n lines of a Pascal's triangle.

Note: A Pascal’s triangle begins with 1 on the top, and 1’s run down the two sides of a triangle. Each new number lies between two numbers and below them, and its value is the sum of the two numbers above it.

Function Prototype: void pascalsTriangle(int n);

Sample: input: 6

output:



Problem 6:

Given a string, remove all adjacent occurrences of the same characters until the string does not have any adjacent duplicate characters.

Function Prototype: char* removeAdjacentDuplicates(char * string);

Sample:

input: azxxzy output: ay
Explanation:
First "azxxzy" is reduced to "azzy".
Now, "azzy" contains duplicates. So it is further reduced to "ay".

input: acaaabbbacdddd, output: acac
explanation: ac(aaa)(bbb)ac(dddd) → acac

input: abccbccba, output: NULL
explanation: ab(cc)b(cc)ba → a(bbb)a → (aa) → NULL