1. In one-dimensional array of integers perform the following operations:
   1. +Delete all even numbers.
   2. +Insert new element after all elements beginning with the indicated digit.
   3. +Delete from array all repeating elements except of their first occurrence.
   4. Insert new element between all element pairs with different signs.
   5. +Compress array by deleting all zero-value elements.
2. In two-dimensional array of integers perform the following operations:
   1. +Insert new line after line containing the first occurrence of the minimal element.
   2. Insert new column before all columns containing the indicated number.
   3. Delete all lines, containing only even number elements.
   4. +Delete all columns, containing only positive elements.
   5. Delete from array the k-th line and the j-th column if their values coincide.
   6. +Compress array by deleting all only zero-value lines and columns.
3. Write an application that provides the following operations with strings:
   1. insert character <x> after every occurrence of character <y>;
   2. mix up the first character with the second one, the third character with the fourth one etc.
   3. find, which of two indicated characters is occurred in the string more often;
   4. count full number of occurrences of <x> and <y> characters;
   5. +count number of different characters in the string;
   6. find out if the string has two adjacent identical characters;
   7. delete the middle character if string length is odd or two middle characters if string length is even;
   8. double every occurrence of the indicated character <x>;
   9. delete all occurrences of the character <x>;
   10. delete all occurrences of the substring <substr>;
   11. +replace all occurrences of the substring <substr1> on the substring <substr2>;
   12. count the sum of all numbers occurred in the string;
   13. count the sum of all digits occurred in the string;
   14. find indexes of the first and the last occurrences of the character <x>;
   15. replace all groups of adjacent dots with ellipsis;
   16. display all characters before the first colon occurrence in the string;
   17. display all characters after the first colon occurrence in the string;
   18. delete all characters inside the parenthesize.
   19. delete all characters inside the curly braces;
   20. count and display statistics of character occurrences in the string.
4. You have message in memory (string) or in text file. Provide the following operations with the message:
   1. Display only words containing the indicated substring.
   2. +Display only words containing not more than n characters.
   3. Display only words beginning with uppercase character.
   4. Display only words containing at least one digit.
   5. Delete from message all words ending with a character <x>.
   6. Delete from message all words containing a character <x> (case insensitive).
   7. Delete from message all one-letter words (with redundant white spaces).
   8. Delete from message all repeating (case insensitive).
   9. Count number of occurrences of the indicated word.
   10. Count number of uppercase words.
   11. Find the longest word in the message.
   12. Find all long (more than 7 characters) words in the message.
   13. Find the shortest word in the message.
   14. Find all short (less than 3 characters) words in the message.
   15. Display all words-palindromes from the message.
   16. Remove all white spaces before comas and ensure that there is white space after every coma.
   17. Display only words occurring just once in the message.
   18. Display only words occurring more than n-times in the message.
   19. Display words from the message in alphabetical order.
   20. Display words from the message in ascending order of their lengths.
5. Misc
   1. Write C# program calculating:
      1. distance between points with indicated coordinates
      2. square and perimeter of triangle by coordinates of its vertices
      3. square of the ring with inner radius r and outer radius R
      4. roots of ax2+bx+c=0 equation
      5. which digit from 4-digit number is greater
      6. display all 2-digit numbers with different digits
      7. display all 3-digit numbers with identical first and third digits
   2. Write a program that inputs a line of text and uses a stack object to print the line reversed.
   3. Write a program that uses a stack to determine whether a string is a palindrome (i.e., the string is spelled identically backward and forward). The program should ignore capitalization, spaces and punctuation.
   4. Write an application that calculates the product of a series of integers that are passed to method product using a variable-length argument list. Test your method with several calls, each with a different number of arguments.
   5. +An integer number is said to be a perfect number if its factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number, because 6 = 1 + 2 + 3. Write method Perfect that determines whether parameter number is a perfect number. Use this method in an application that determines and displays all the perfect numbers between 2 and 1000. Display the factors of each perfect number to confirm that the number is indeed perfect.
   6. Write recursive method Power( base, exponent ) that, when called, returns baseexponent.

For example, Power(3, 4) = 3 \* 3 \* 3 \* 3. Assume that exponent is an integer greater than or equal to 1. Hint: The recursion step should use the relationship

base exponent = base · base exponent -1

and the terminating condition occurs when exponent is equal to 1, because base1 = base

Incorporate this method into an application that enables the user to enter the base and exponent.