Mandatory tasks

1. Print the number of arguments to the console. Explain why this number is never less than 1.
2. Write a program that multiplies the "first" argument by the "second" argument. Use addition inside a loop to achieve this. Write the result to the console.
3. Write a recrusive function that calculates the factorial of a given number.
4. Write a program that stores our name in a "player.txt" file.
5. Write a program, which writes the even elements of an integer array to a file called "even_numbers.txt". Run this program several times, and explain why the file remains the same.
6. Modify the previous program such that every time the program runs, the result of the program appends to the file contents.
7. Write a new program that reads the file created by the excercise 6, and adds the numbers (of the file) together. Write the output on the console.
8. Write a search algorithm using a do-while loop. Return the index where the given element can be found. If the given element is found, stop the loop (using 'break').
9. Write a 'toLowerCase' function. the function should get a character array, and each capital letter (you can find determine whether a character is a capital letter, using the ASCII table) and change it to its equivalent lower-case letter! After this, output to the console which letter was changed. If a character does not need to be changed, then skip that element (using 'continue').
10. Lets write a LOG-INFO-WARNING-ERROR function. The first parameter should be a positive integer, which indicates the nature of the printed message (i.e.: 0-INFO, 1-WARNING, 2-ERROR, everything else is LOG). The second parameter should be the message. Using switch-case, determine what should be printed before our message. (i.e.: "LOG - user signed in.", "ERROR - could not connect to server."). Demonstrate what happens if we do not use the "break" inside each case. Demonstrate what happens if we do not write a default branch.

Optional tasks

1. Write a program that puts arbitrary data inside a file. Before closing the file, block the program with a blocking function (i.e.:scanf). Examine the contents of the file that we wrote before closing it. Let the program continue, and examine the contents of the file again.
2. Write a program that reads strings from the standard input, or from a file. For the next excercises use string functions. •Write words that are longer than 5 characters to the standard output. •Write words that contain a given

letter to the standard output. •Write the words that contain 'apple' to the standard ouput. (i.e.: 'applepie') •Write words that contain numbers to the standard output.

3. Write a program that prints out the n-th row of a file. Use the command line arguments to specify the name of the file and the value of n.

Advanced tasks

1. Write a program that counts the number of different words in a file. Define a set-like implementation using an array.

2. Write a program, which translates the content of a file to Morse-code. Write the encoded message in a new file. The names of the input and output file should be read from the program arguments.