Project **CALCULATOR**

This project consists of three quite separate sections:

1. **The button section**. There will be:
   * all the digits (of course!);
   * the four operations, and the brackets;
   * the percentage sign; (optional – should I really implement it?)
   * the “del” button, which deletes a single operation sign, a digit, a bracket or an otherwise represented operation (say log, ln or sin);
   * the “del2” (name unknown) button, which deletes a single operation sign, a number, an open bracket, the closest piece of expression in brackets if used after a -paired- closed bracket, or an otherwise represented operation (say log, ln or sin);
   * the “erase” button, which erases the whole expression written so far.
   * the “equals” button, of course. This one is tricky: not only will it trigger the computation of the expression written so far (which makes the computation section come into action), it will also trigger a series of graphic events (which makes the graphical section come into action). Specifically, while the computation section calculates the output, the graphical section will move the expression to the top, shrinking it and rendering it in grey, while the output will be displayed in white, bigger than the expression, at the bottom, exactly how the expression was displayed before.
   * anything more? I don’t think so.
2. **The graphical section**. There will be:
   * The buttons, obviously;
   * Two input/output lines. The input will be displayed on the bottom line at first, then the expression will be displayed on the top line instead. The outcome will be displayed on the bottom line.
   * If a button is pressed after the output is displayed, this will erase the previous expression and start a new one with the aforementioned button, but the output will remain in place, on the bottom line. If the inserted character is deleted before any further button is pressed, then the program will revert the previous equation as if nothing happened. Of course, the characters inserted after a button is pressed vary depending on the operation represented by that button (although this is something that will only come in handful in the making of the advanced level).
   * A history button.
3. **The computation section**. This is undoubtedly the hardest part. There will be:
   * La parte principale, che analizza la stringa di input. Un’opzione è la seguente:
     1. Leggere la stringa riconoscendo ogni numero (dove per numero si intende una o più cifre, comprese fra due simboli).
     2. Riconoscere un numero prima o dopo una parentesi come moltiplicazione.
     3. Salvare il risultato come una stringa.
     4. Creare un file di testo e inserire la stringa (senza virgolette, ovviamente!)
     5. Compilare il file e salvare il risultato.
     6. Restituire il valore così ottenuto come double.
   * Un’altra opzione, ben più macchinosa, è questa:
     1. Assegnare una variabile al numero di parentesi aperte.
     2. pass