The printf procedure is a procedure present in many programming languages allowing to make a formatted impression. A formatted print is the print of a text, called the model, inside which certain sequences, called the labels, are replaced by the values supplied to printf. For example, calling printf("%s is a %s", ["The tomato", "fruit"]) will print "The tomato is a fruit".

For this assignment, you must write the printf procedure. This procedure has two parameters. The first parameter is a text representing the model. The second parameter is an array of texts used to populate the template. The procedure returns no value.

The printf procedure replaces the labels made up of "%s" characters in the model by the elements contained in the array provided as the second parameter. The first occurrence is replaced by the first element of the array, the second occurrence by the second element, and so on. The result of filling the model is then printed in the console with print.

Since the "%" character has a special meaning in the pattern, it must be treated in a special way. It must be followed by the character "s". However, to be able to print the "%" character, it is allowed to escape a "%" by prefixing it with another "%". For example, calling printf("This homework is worth %s%%", ["2.5"]) will print "This homework is worth 2.5%". An unescaped "%" followed by any character other than "s" or "%" is therefore invalid.

A call to printf may be invalid. This is the case if an unescaped "%" character is followed by any character other than "s" or "%". The call is also invalid if the number of labels in the pattern is not equal to the length of the array (the second parameter). In case of invalid call, the printf procedure should not print anything and should throw an error. You can throw an error using the raise RuntimeError("invalid format") statement. This statement will terminate the program and display an error message in the console.

For this assignment, only the printf procedure is required. However, you can write auxiliary functions as needed. Since the printf procedure returns no value, you don't have to write any unit tests. However, you are encouraged to choose a good functional decomposition and write unit tests for your helper functions.

Your program only needs to define the printf procedure, plus auxiliary procedures as needed. It must not execute any procedure or function calls. Your code should also not do any input-output (no call to input, alert, or prompt). You must imagine that your code will be used by another programmer in another software.