Problem 1/2 (Unique):
Create a program, using Python or Go, to identify the unique tokens in a file. The program shall prompt the user for a file name at program execution. The program shall read ASCII strings from the file until it reaches the EOF symbol. From the accumulated input, the application shall then extract only its unique tokens, separated by the space character, and write these to a new output file. For example, if provided with a file containing a text document, it shall write to the output file, with white-space between each token, every \textit{unique} word contained therein. The order the words as they appear in the output does not need to be sorted in any particular order.

Problem 2/2 (Crunch):
Create a program to randomly combine four tokens found in an input file producing an output token with the four words squeezed together (e.g., somewhatlikethisbutrandom). It shall prompt the user for a file name at program execution. Using the name provided at run-time, the program shall read in the entire set of words to use during the combination process. After reaching the EOF symbol, the program shall then begin randomly selecting words and combining them into a concatenated output word. The program produces only one crunched output word for each execution. There shall be no duplicate words in any single output token (although combining 'racecar' with 'car' could make it appear so erroneously). The program may ignore words in the input file with a length less than four.