https://github.com/danharangus/FLCD

SymbolTable - Based on a hash table
Methods: - insert(self, name) - inserts the symbol with the given name in the table and returns the position (or returns the position if it already exists)
- get(self, name) - gets the index of the symbol in the symbol table, or None if it doesn't exist

HashTable - Implemented using open addressing
Methods: - _init_(self, initial_size, load_factor) - creates a new HashTable. Default size is 10 and default load factor is 0.7
- resize(self, new_size) - resizes the hash table
- hash_function(self, key, size) - hash function based on the actual hash table size
- put - inserts a key-value pair in the hash table and automatically doubles the size should it be the case (using the resize method)
- get - gets the value of a key from the hash table

PIF - for the PIF, I use a pairs array, where a pair is of the form (token, pos) where token is the token itself or the code 'id' if it's an identifier or the code 'constant' if it's a constant, and pos is the position in the symbol table or -1 tokens that are not identifiers or constants
Methods - add(token, position)

Scanner - class which handles the scanning of a code file
Methods - Scanner(program_file_name, token_file_name) - constructor
- scan -> method which scans the program file and returns the pif, the constants symbol table and the identifiers symbol table

For the scanner, I used the following regular expressions:
string constants: ^[a-zA-Z0-9_ ?\-:*^+=.!]*$ -> will match any string which contains small or capital letters or digits, or the special characters ?, :, *, ^, +, =, space, _, ., ! and - (which had to be escaped) in any order and any number of times

char constants: ^[a-zA-Z0-9_ ?\-:*^+=.!]$ -> same as string constant, but there can only be one character

int constants: ^(0|[-]?[1-9][0-9]*)$ -> any digit from 1 to 9 which can be optionally preceded by a "-" and other digits from 0 to 9 (any number of them)

identifiers: ^[a-zA-Z]([a-zA-Z]|[0-9])*$ -> any string that starts with a letter and is followed by any number of letters or digits