https://github.com/trutadan/University-Work/tree/main/Semester%205/Formal%20Languages%20and%20Compiler%20Design

The SymbolTable class is a data structure designed for managing symbolic names, such as identifiers and constants, encountered during the compilation of source code. It is a part of the symbol table system and provides methods for adding, searching, and removing symbolic names. The symbol table utilizes a hash table as its underlying data structure, but we will focus on the symbol table implementation, since the 2 of them are almost the same.

1. public String getTerm(Pair<Integer, Integer> position) throws IndexOutOfBoundsException
    - Parameters: position - A pair of integers representing the position of a symbolic name in the symbol table.
    - Returns: The symbolic name at the specified position.
    - Throws: IndexOutOfBoundsException - If the specified position is invalid.
    - Description: Retrieves the symbolic name at the given position in the symbol table.

2. public Pair<Integer, Integer> search(String term)
    - Parameters: term - The symbolic name to search for.
    - Returns: A pair of integers representing the position of the symbolic name in the symbol table, or (-1, -1) if not found.
    - Description: Searches for the specified symbolic name in the symbol table and returns its position.

3. public boolean exists(String term)
    - Parameters: term - The symbolic name to check for existence.
    - Returns: true if the symbolic name exists in the symbol table, false otherwise.
    - Description: Checks whether the specified symbolic name exists in the symbol table.

4. public Pair<Integer, Integer> add(String term)
    - Parameters: term - The symbolic name to add to the symbol table.
    - Returns: A pair of integers representing the position of the added symbolic name.
    - Description: Adds the specified symbolic name to the symbol table and returns its position. If the name already exists, it returns the existing position.

5. public void remove(String term) throws NoSuchElementException
    - Parameters: term - The symbolic name to remove from the symbol table.
    - Throws: NoSuchElementException - If the specified term is not found in the symbol table.
    - Description: Removes the specified symbolic name from the symbol table. If the term is not found, it raises an exception.

The symbol table is represented by a hashtable. The hashtable is implemented using a list of lists. The hashtable also has a size. An element from the table has as its position a pair of indices, the first one representing the index of the list in which the element should be stored(you can call it row, if you view it as a matrix), and the second one being the position in that list. As suggested in the course, the hash function was computed as the sum of the ASCII codes of the characters modulo the size of the list.

The pair mentioned above is another class, which also has its own implementation. It is quite intuitive and straightforward. The purpose of it is to store two related pieces of data together. It has a generic structure, meaning that any kind of data type can be used in a pair. Even so, in our symbol table implementation, we were only interested in a pair of integers representing the position of an element in the table. The 'valid' method checks if both the key and value are of type Integer and, if so, whether they are both equal to -1. If both conditions are met, it returns false (indicating the pair is not valid); otherwise, it returns true. In other words, the function checks if the pair represents a possible position in the table or not.