https://github.com/Arnold-21/Compiler

In the finite automata the states, final states and the alphabet is stored as a vector of strings, the initial stri
ng is stored as a string.
The transitions are stored as a map from the combination of the from state and transition variable to the th
e next state.
The decision of storing as a map, is quicker compraison, since to check if a transition is valid, we only nee
d to check if the specific key exists in the map.
These parameters are from a file, which describes a finite automata.

The two function of the automata are the following:
1. CheckWord
- Given a string word, the automata starts from an initial position, empty position, and for every letter it checks if there exists a transition from the current state with the letter to another,
if there exists one, it updates the state, if not it returns false, since the word is not accepted
- After every letter has been checked the final check decides if the word is accepted or not, by checking if the current state is in the final states of the automata

2. GetAccepted
- Works with the same logic as the previous function, with the difference that when a new state is not foun
d, the loop is broken and the word is built up by the
letters that are checked
- At the end, we check the final state again, and instead of true, we return the built word
- This gives back an accepted word from the beggining of a sequence, if there is one

----
EBNF forms of the in files

1. Identifier:

```
letter = "A" | "B" | ... | "Z" | "a" | ... | "z".
digit = "0" | "1" | ... | "9".
identifier = letter{letter | digit}.
```

2. Integer:

```
digit = "0" | "1" | ... | "9".
non_zero_digit = "1" | "2" | ... | "9".
int = ["+" | "-"]non_zero_digit{digit} | "0".
```