AFP - Assignment 1 - Stack Permutation

Maximo Garcia Martinez

1. Algorithm

1. Reverse both Input (I) and Output (O) lists. Create a new empty list called Stack (S).
2. If I, O and S are empty return true.
3. If the head of S and the top of O are equals:
   - If S and I are empty and O is not, return false.
   - Otherwise, execute the algorithm again from step 3, with I, the tail of S and the tail of O.
4. If the head of I and the top of O are equals:
   - If S and I are empty and O is not, return false.
   - Otherwise, execute the algorithm again from step 3, with S, the tail of I and the tail of O.
5. Append the head of I into S. Go to step 3.

2. Testing

I have created a file called `test_cases.txt` that it can be found in my submission. Some lines of this file are:

```
{[1,2],[2,1],true}.
{[2,1],[2,1],true}.
{[3,1,2],[3,2,1],true}.
{[1,2,3,4],[2,4,1,3],false}.
{[1,2,3,4],[2,1,4,3],true}.
```

The test function reads all the lines from this files and iterate for each case checking if L2(Output) can be a stack permutation of L1 (Input). The code used for this is shown below.

```
perm_test_() ->
    {ok, Cases} = file:consult("test_cases.txt"),
    [?_assertEqual(O, (perm(L1, L2))) || {L1, L2, O} <- Cases].
```

I have two other function for test the speed of the algorithm with one million elements. Both fucntions are shown below.

```
perm_million_test() ->
    [?_assertEqual(true, (perm:perm(L, L))) || L <- lists:seq(1, 1000000)].

perm_reversemillion_test() ->
    [?_assertEqual(true, (perm:perm(L, L))) || L <- lists:reverse(lists:seq(1, 1000000))].
```