Homeworks — Stack

(1) Add to the parenthesization checking program curly brackets '{' and '}'. Check whether you can deal with the following expressions:

(1) `( ( [ ] { ( ) } ) )` good

(2) `( { [ ] } )` good

(3) `( ) ( ] { }` bad

(2) Currently, the Polish notation program does not check whether at the end of processing, there are still more values left on the stack. Also, we can implement exponentiation as well.

(1) Deal with this keeping the stack interface as is. (After popping the result, the stack should be empty).

(2) Deal with this by adding a function to the stack that returns the length of the list in the stack.

(3) Implement exponentiation as an operator.

(3) Write a function that reverses a string by pushing and popping its letters on a stack. (I know that you can do so in many other ways, including slicing or using the reverse function, but this exercise asks you to do it with a stack.)

(4) In Unix, files are organized into directories. Paths lists the directories going up and down the directory trees, either from root or from the current directory, in which case this is a relative path. The directory names are separated by a forward slash '/' . When specifying a relative path to a file, it is possible to use the single dot . to denote the current directory and two dots to denote the parent directory. This means that the same path can be written in different ways. For example:
`../../t1/t2/../file` is the same as `../../t1/file`
`.././t1/t2/../file` is the same as `../t1/file`
`t1/../../t2/file` is the same as `../t2/file`
Write a program that takes a relative path and removes as much as possible the double dots and the single dots. You should use a stack on which you push the directory names (and the file name at the end). Whenever you are about to push a single dot, you leave it out and whenever you are about to push a double dot, you see whether the stack is empty, in which case you just pop what was there as the double dot and the directory name cancel each other out. `dir1/../` just means going down to `dir1`, and the to go up, so we can save ourselves this trip.