Sanitizing Pathnames (flex)

In Posix pathnames, *components* are separated by `/`. Consecutive multiple `/` have the same meaning as a single `/`. A final `/` has no meaning, but an initial `/` is significant. Leading and trailing spaces are allowed but have no significance. A component consists of `a-z`, `A-Z`, `0-9`, and `.` (dot), with two special cases: a component with a single `.` component refers to the current directory and a `..` component refers to the parent directory. Note that `.` can also be part of a component. Portable pathnames restrict each component to having at most 14 characters, and the whole pathname can have at most 255 characters.

Implement a sanitizer for pathnames using Flex and C. Your implementation should read from standard input and produce a sanitized portable pathname on standard output or an error message on standard error. The implementation has to use the regular expression facilities of flex to check for the well-formedness of the input.

standard input	standard output
`/aaa//bb/c/`	`/aaa/bb/c`
`aaa/b.b/../cc/./dd`	`aaa/cc/dd`

The sanitizer should read the input line by line from standard input until the end of the file. For each line, the sanitizer should either produce one line with the sanitized pathname on standard output or an error message on standard error and terminate immediately:

standard input	standard error
`/a//b/#/c`	`invalid character`
`/012345678901234/bb`	`component too long`
`aa/../..`	`malformed pathname`
`/this/is/a/path/name/that/is/really/too/long/.../way/too/long/`	`pathname too long`

Hint: use the regular expression features of flex to check for invalid characters, too long components, and to "swallow" leading and trailing spaces, multiple consecutive `/`, and `.` components.

```
%%writefile spn.l
%option noyywrap
%{
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

# YOUR CODE HERE
raise NotImplementedError()
%}
# YOUR CODE HERE
raise NotImplementedError()
```

```
Writing spn.l
```

```
!flex spn.l

!cc -o spn -std=c99 lex.yy.c -D_POSIX_C_SOURCE=1
```

The file `goodpaths.txt` contains a set of paths to test. The extra newline is needed as the `%%writefile` trims a trailing newline if it is at the end of the input.

```
%%writefile goodpaths.txt
/aaa//bb/c/
aaa/b.b/../cc/./dd
a45/b.b/../cc/./dd/.
./////def/ghi///jkl//mno/pqr/../.././../../ghi/./jkl////
/.../.abc/./123/456/789/../../
./test/ing

%%capture output
!cat goodpaths.txt | ./spn
# Should output
#/aaa/bb/c
#aaa/cc/dd
#a45/cc/dd
#def/ghi/jkl
#/.../.abc/123
#test/ing
#

print(output) # for testing purposes

expected =  """/aaa/bb/c\r
aaa/cc/dd\r
a45/cc/dd\r
def/ghi/jkl\r
/.../.abc/123\r
test/ing\r
\r
"""
actual = str(output)
# Use these outputs to help debug line endings if needed
print(repr(actual))
print(repr(expected))
assert actual == expected

%%capture output
!echo "/a//b/#/c" | ./spn # Should output `invalid character`

print(output) # for testing purposes

assert str(output) == 'invalid character\r\n'
```

```
%%capture output
!echo "/012345678901234/bb" | ./spn # Should output `component too
long`

print(output) # for testing purposes

assert str(output) == 'component too long\r\n'

%%capture output
!echo "aa/../.." | ./spn # Should output `malformed pathname`

print(output) # for testing purposes

assert str(output) == 'malformed pathname\r\n'

%%capture output
#long_path = '/'.join(['0123456789' for _ in range(26)])
long_path = '/'.join(['0123456789' for _ in range(26)]) +
'/'.join(['..' for _ in range(23)])
!echo $long_path | ./spn # should output `pathname too long`

print(output) # for testing purposes

assert str(output) == 'pathname too long\r\n'

%%capture output
!echo "/abc/123/abcdefghijklmno" | ./spn # Should output `component
too long`

print(output) # for testing purposes

assert str(output) == 'component too long\r\n'

%%capture output
!echo "/abc/def\xE2\x98\xA0/" | ./spn # Should output `invalid
character`

print(output) # for testing purposes

assert str(output) == 'invalid character\r\n'

%%capture output
!echo "abcdef/./def/../../.." | ./spn # Should output `malformed
pathname`

print(output) # for testing purposes

assert str(output) == 'malformed pathname\r\n'
```