Challenge #1

From Perl Weekly Challenge - 012 retrieved on 2019-06-15 at 20:51 +00:

The numbers formed by adding one to the products of the smallest primes are called the Euclid Numbers (see wiki). Write a script that finds the smallest Euclid Number that is not prime. This challenge was proposed by Laurent Rosenfeld.

The Perl 6 program

```perl
# Perl Weekly Challenge - 012
# Challenge #1
#
# See
# engineering.purdue.edu/~mark/pwc-012.pdf
# for more information.

# Run using Perl 6.
use v6;

# Get prime numbers from 2 to 1,000.
# This doesn't read left-to-right
# my @prime = grep &is-prime, (2..1_000);
# as well as this does:
my @prime = (2..1_000).grep(&is-prime);

for (^@prime.elems) -> $i
{
    # This could be optimized by keeping a running product of the
    # first n primes and then just multiplying by the (n+1)st prime
    # but
    # Premature optimization is the root of all evil.
    # ---Donald Ervin Knuth
    my $e = ([*] @prime[0..$i]) + 1;
    ($e.is-prime) or $e.say, last;
}
```

1
Challenge #2

From Perl Weekly Challenge - 012 retrieved on 2019-06-15 at 20:56 +00:

Write a script that finds the common directory path, given a collection of paths and directory separator. For example, if the following paths are supplied.

/a/b/c/d
/a/b/cd
/a/b/cc
/a/b/c/d/e

and the path separator is /. Your script should return /a/b as common directory path.

Discussion

Start with my standard comment block and “use v6” for Perl Weekly Challenge problems:

# Perl Weekly Challenge - 012
# Challenge #2
#
# See
# engineering.purdue.edu/~mark/pwc-012.pdf
# for more information.

# Run using Perl 6.
use v6;

Define @path and $sep. Include some other @path values for debugging.

# Define @path and $sep.
my Str @path;
# Possible @path test lines.
@path = <</a/b/c/d /a/b/cd /a/b/cc /a/b/c/d/e>>; # correct answer is "/a/b"
# @path = <</a /b>>; # correct answer is ""
# @path = <</a/b /a/c>>; # correct answer is "/a"
# @path = <</a/b/c/d /a/b/c/d /a/b/c/d>>; # correct answer is "/a/b/c"
my $sep = '/';

Error check @path in the development version of this program. This won’t be in the final version of this program.

# Error check @path.
for @path
{
    (/^-$sep/) or die qq/path "${_}" does not start with "$sep"/
    (/^$sep$$/) and die qq/path "${_}" ends with "$sep"/
}

Split @path into its separate parts and put the results in the two-dimensional @part array. This includes several alternatives for doing this that won’t appear in the final program.
# Split @path into its separate parts and put
# the results in the two dimensional @part array.

my @part;

# This
#   (0..^@path.elems).map({@part[$_] = split(/$sep/, @path[$_]})
# or
# loop (my $i = 0; $i < @path.elems; $i++)
# {  
#     @part[$i] = split(/$sep/, @path[$i]);
# }
# or
# for @path -> $path
# {
#     push @part, split(/$sep/, $path);
# }
# or
# for @path
# {
#     push @part, split(/$sep/, $_);
# }
# or
# push @part, split(/$sep/, $_) for @path;

Compute and print the output.

# Compute index of last directory part of @part with fewest number of parts.
my $lastindex = @part.map({.elems}).min - 2;

# Set $index to one past the last directory part that matched.
# Search for unequal directory parts or ``fall out the bottom''
# of the loop if all directory parts match for this $index.
my $index = 0;
while ($index <= $lastindex && [eq] @part[*;$index])
{
    $index++;
}

# Print the output.
say join '/', @part[0;^$index];
The Perl 6 program

```perl
# Perl Weekly Challenge - 012
# Challenge #2
#
# See
#   engineering.purdue.edu/~mark/pwc-012.pdf
# for more information.

# Run using Perl 6.
use v6;

# Define @path and $sep.
my Str @path = <</a/b/c/d /a/b/cd /a/b/cc /a/b/c/d/e>>;
my $sep = '/';

# Split @path into its separate parts and put
# the results in the two-dimensional @part array.
# 0 1 2 3 4 5
# 0 a b c d
# 1 a b cd
# 2 a b cc
# 3 a b c d e
my @part;
for @path
{
    push @part, split(/$sep/, $_);
}

# Compute index of last directory part of @part with fewest number of parts.
my $lastindex = @part.map({.elems}).min - 2;

# Set $index to one past the last directory part that matched.
# Search for unequal directory parts or ``fall out the bottom''
# of the loop if all directory parts match for this $index.
my $index = 0;
while ($index <= $lastindex && [eq] @part[*;$index])
{
    $index++;
}

# Print the output.
say join '/', @part[0;~$index];
```