

Forth Programming Challenge Solutions

September 26, 2020

By Joel Shprentz, jshprentz@his.com

Forth: gforth

System: Raspberry Pi 4

alphabet.fs Sun Sep 27 00:30:30 2020 1

\ Challenge 1: Print the alphabet "AbCdEfGh . . . z" in alternating case
\ using three methods.

\ Method 1: Loop through a-z, converting odd letters to upper case.

```
: ?odd ( c -- f ) 1 and ;  
: toupper-odd ( c -- c | C ) dup ?odd if toupper then ;  
: from-to ( n1 n2 -- n3 n1 ) 1+ swap ;  
: alphabet1 'a 'z from-to ?do i toupper-odd emit loop ;
```

alphabet1 cr

\ Method 2: Advance from upper to lower case letters and from lower to upper case.

```
: upper-to-next-lower ( C1 -- c2 ) 'b + 'A - ;  
: lower-to-next-upper ( c2 -- C2 ) 'C + 'b - ;  
: alphabet2  
  'A begin  
    dup emit upper-to-next-lower  
    dup emit lower-to-next-upper  
    dup 'Y >  
  until  
  drop ;
```

alphabet2 cr

\ Method 3: Display the alphabet string.

```
: alphabet3 ." AbCdEfGhIjKlMnOpQrStUvWxYz" ;
```

alphabet3 cr
bye

AbCdEfGhIjKlMnOpQrStUvWxYz
AbCdEfGhIjKlMnOpQrStUvWxYz
AbCdEfGhIjKlMnOpQrStUvWxYz

gcd.fs Fri Sep 25 00:32:17 2020 1

\ Challenge 2: Print a table headed by 240 to 249 across with 170 to 179 down.
\ At each intersection show the Greatest Common Divisor.

```
: is-odd? ( u -- f ) 1 and ;
defer gcd
: gcd-odd-even ( u-odd u-even -- u ) 2/ gcd ;
: gcd-both-even ( u1 u2 -- u3 ) 2/ swap 2/ gcd 2* ;
: gcd-both-odd ( u1 u2 -- u3 ) 2dup > if swap then over - gcd ;
: gcd-top-odd ( u1 u2 -- u3 )
  swap dup is-odd? if gcd-both-odd else gcd-odd-even then ;
: gcd-top-even ( u1 u2 -- u3 )
  over is-odd? if gcd-odd-even else gcd-both-even then ;
: gcd-soon ( u1 u2 -- u3 )
  2dup = if drop else
  dup 0= if drop else
  over 0= if swap drop else
  dup is-odd? if gcd-top-odd else
  gcd-top-even
  then then then then ;
' gcd-soon is gcd
: from-to ( n1 n2 -- n3 n1 ) 1+ swap ;
: row-range ( -- i2 i1 ) 170 179 from-to ;
: column-range ( -- i2 i1 ) 240 249 from-to ;
: header. ( -- ) cr 3 spaces column-range ?do i 4 u.r loop ;
: gcd-row. ( u -- u ) column-range ?do dup i gcd 4 u.r loop ;
: row. ( u -- ) cr dup 3 u.r gcd-row. drop ;
: table. ( -- ) header. row-range ?do i row. loop ;
table.
cr bye
\ vim: tabstop=8 expandtab shiftwidth=4 softtabstop=4 autoindent
```


primes.fs Sun Sep 27 00:19:17 2020 1

\ Challenge 3: Print the five largest prime numbers less than 10,000.

10000 constant PRIME-LIMIT
5 constant TOP-N

```
: :flags ( n "name" -- addr )  
  create here over allot swap erase here  
  does> ( n -- addr ) + ;
```

PRIME-LIMIT :flags sieve
constant MEM-LIMIT

```
: square ( n -- n ) dup * ;  
: mark-non-primes ( prime -- ) MEM-LIMIT over square sieve ?do true i c! dup +loop drop ;  
: next-prime ( n incr -- prime ) swap begin over + dup sieve c@ 0= until nip ;  
: next-higher-prime ( n -- prime ) 1 next-prime ;  
: next-lower-prime ( n -- prime ) -1 next-prime ;  
: sieve-done? ( n -- f ) square PRIME-LIMIT >= ;  
: fill-sieve ( -- )  
  2 begin  
    dup mark-non-primes  
    next-higher-prime  
    dup sieve-done?  
  until drop ;  
: top-primes. ( -- ) PRIME-LIMIT TOP-N 0 ?do next-lower-prime dup . cr loop drop ;  
: primes ( -- ) fill-sieve top-primes. ;
```

primes
bye

\ vim: tabstop=8 expandtab shiftwidth=4 softtabstop=4 autoindent

primes.out

Sun Sep 27 00:31:25 2020

1

9973
9967
9949
9941
9931

```
\ Challenge 4: Using numbers, terminal characters or graphics plot x-squared  
\ from 1 to 10.
```

```
: plot-square ( n -- ) dup 2 u.r dup * 2/ 1+ spaces '* emit cr ;  
: plot-squares ( -- ) 11 1 ?do i plot-square loop ;  
: y-axis ( -- ) 3 spaces 10 0 ?do ." +----" loop '+ emit cr ;  
: y-labels ( -- ) 101 0 ?do i 4 u.r space 10 +loop cr ;  
: y-title ( -- ) 23 spaces ." x squared" cr ;  
: x-title ( -- ) ." x" cr ;  
: squares ( -- ) x-title plot-squares y-axis y-labels y-title ;
```

```
squares
```

```
bye
```

```
\ vim: tabstop=8 expandtab shiftwidth=4 softtabstop=4 autoindent
```


