Loop

Όλα τα προγράμματα έχουν γραφεί με την υπόθεση ότι input του προγράμματος είναι στις μεταβλητές i1, i2, ... και output στις μεταβλητές o1, o2, ... για να επιβεβαιωθεί η ορθότητα τους σε έναν διερμηνέα.

Σχόλια θεωρούνται γραμμές που ξεκινούν με "--"

1.

mult(x, y)

```
y := i1;
z := i2;

x := 0;
for w := 1 to z do
        x := add(x, y);
done
o1 := x;
```

sub(x, y)

```
x := i1;
y := i2;
for w := 1 to y do
        x := x - 1;
done
o1 := x;
```

ifzero(x, y)

```
x := i1;
y := 1;
for w := 1 to x do
        y := y - 1;
done
o1 := y;
```

2.

if then else

```
z := g(x);
y := ifzero(z);
for w := 1 to y do
        h1(x);
done

y1 := ifnzero(z);
for w := 1 to y1 do
        h2(x);
done
```

3.

div(x, y)

```
x := i1;
y := i2;

-- if divisor = 0 => output = 0
z := ifzero(y);
for w := 1 to z do
        o1 := 0;
done

z := ifnzero(y);
r := 0;
-- if divisor != 0 => division
for w := 1 to z do
        x1 := x;

        for i := 1 to x do

                x2 := greater(x1, y);
                x3 := equals(x1, y);
                x2 := or(x2, x3);
-- if >= subtract && increment counter r
                for j := 1 to x2 do
                        x1 := sub(x1, y);
                        r := r + 1;
                done

        done
        o1 := r;
done
```

4.

x^y - pow

```
x := i1;
y := i2;

x1 := 1;
for w := 1 to y do
        x1 := mult(x1, x);
done
o1 := x1;

-- this can be faster by x <- x * x;
```

$n!$

$0! = 1, n! = n * (n-1)!$

```
x := i1;
z := 1;
for w := 1 to x do
        z := mult(z, w);
done

o1 := z;
```

$\binom{n}{a}$

$\binom{n}{a} = \frac{n!}{(n-a)!a!}$

```
n := i1;
k := i2;
n1 := factorial(n);
k1 := factorial(k);
d := sub(n, k);
d := factorial(d);
d := mult(d, k1);
o1 := div(n1, d);
```

$\lfloor \sqrt{n} \rfloor$

Ο αλγόριθμος είναι $\mu w((w+1)^2 > x)$ το οποίο όμως φράζεται από το $x/2$ άρα υπολογίζεται με μία for loop ως εκεί.

```
x := i1;
y := 1;
y := y + 1;
x1 := div(x, y);
z := 1;
r := 0;
for w := 1 to x1 do
        k1 := w + 1;
        k1 := mult(k1, k1);

-- w^2 + 1 > x => w is the result but only the first time
        b := greater(k1, x);
        b := and(b, z);
        for l := 1 to b do
                r := w;
-- flip z to 0 because it acts as a boolean the first
-- time we exceed x
                z := 0;
```

```
        done
done
o1 := r;
```

Βοηθητικά προγράμματα που χρησιμοποιήθηκαν στα παραπάνω

greater (>)

```
x := i1;
y := i2;
d := sub(x, y);
o1 := ifnzero(d);
```

and

```
x := i1;
y := i2;
o1 := 0;
x1 := ifnzero(x);
for w := 1 to x1 do
        x2 := ifnzero(y);
        for i := 1 to x2 do
                o1 := 1;
        done
done
```

5.

Αρχικά θα υπολογίσουμε την βοηθητική mod

mod(x, y)

```
x := i1;
y := i2;
z := ifzero(y);
-- if y == 0 -> return 0
for w := 1 to z do
        o1 := 0;
done
z := ifnzero(y);
-- if y != 0 -> return (x - y * div(x, y))
for w := 1 to z do

        q := div(x, y);
        p := mult(y, q);
        r := sub(x, p);
        o1 := r;
done
```

i. divisible(m, n)

```
-- if mod(x, y) == 0 => return true else false
m := i2;
n := i1;
z := ifnzero(n);

for w := 1 to z do
        m := mod(m, n);
        o1 := ifzero(m);
done
z := ifzero(n);
for w := 1 to z do
        o1 := 0;
done
```

ii. prime(n)

```
n := i1;
n1 := sqrt(n);
z := 0;
for w := 1 to n1 do
        w1 := w + 1;
        k := divisible(w1, n);
        for i := 1 to k do
                z := 1;
        done

done
o1 := ifzero(z);
```

iii. p(n) - nthprime(n)

```
-- find an upperbound
-- for the nth prime to count to that
-- p(n) -> nth prime
-- p(n+1) -> [p(n)+1, p(n)!+1]
-- p(1) = 2.
-- p(n+1) -> least prime in [p(n)+1, p(n)!+1]
```

```
n := i1;
n := n-1;

-- z = 2; the first prime
z := 1;
z := z + 1;


for w := 1 to n do

        l := z + 1;
        u := factorial(z);
        u := u + 1;
        d := sub(u, l);
        f := 0;
        for i := 0 to d do
                t := add(i, l);
                t1 := prime(t);
                t2 := ifzero(f);
                t2 := and(t1, t2);
                for j := 1 to t2 do
                        f := 1;
                        z := t;
                        d := 0;
                done
        done
done
o1 := z;

-- can also
-- hold the product of primes less than the current prime
-- and keep that + 1 instead of p(n)!+1
```