Μοντέλα Υπολογισμού
LOOP
Στάθης Ζάχος

1. Γράψε ένα τυπικά ορθό πρόγραμμα LOOP για πολλαπλασιασμό ($\mathrm{mult}(x, y)$), αφαίρεση ($\mathrm{sub}(x, y)$) και $\mathrm{ifzero}(x)$.

2. Γράψε ένα πρόγραμμα LOOP για το

   $f(x) := \textbf{if } g(x) = 0 \textbf{ then } h_1(x) \textbf{ else } h_2(x)$

   όπου $g(x)$, $h_1(x)$, $h_2(x)$ ήδη ορισμένα LOOP προγράμματα.
   Σημείωση: Το πιο πάνω πρόγραμμα μπορείτε στην συνέχεια, αφού το ορίσετε, να το χρησιμοποιείτε.

3. Γράψε ένα πρόγραμμα LOOP για διαίρεση

$$\mathrm{div}(x, y) := \begin{cases} x \textbf{ div } y, & y \neq 0 \\ 0, & y = 0 \end{cases}$$

4. Γράψε προγράμματα LOOP για x^y, $n!$, $\binom{n}{a}$, $\lfloor \sqrt{n} \rfloor$.

5. Γράψε προγράμματα LOOP για

   i. $\mathrm{divisible}(m, n)$, (διαιρεί ο n τον m;)
   ii. $\mathrm{prime}(n)$ (είναι ο n πρώτος;)
   iii. $\mathrm{p}(n)$ (επιστρέφει τον n-οστό πρώτο)

Κεφάλαιο 3

1.

***mult*(*x*, *y*)**

```
z:=0;
for i:=1 to y do
        for j:=1 to x do
                z:=z+1;
        end;
end;
return z;
```

***sub*(*x*, *y*)**

```
z:=x;
for i:=1 to y do
        z:=z-1[1];
end;
return z;
```

***ifzero*(*x*)** [2]

```
z:=x;
w:=x-1;
for i:=1 to w do
        z:=z-1;
end;
return z;
```

2.

```
w:=g(x);
z:=h1(x);
//Αν το w είναι 0, δεν θα εκτελεστούν οι εντολές εντός της επανάληψης, οπότε η μεταβλητή z θα έχει τιμή h1(x)
for i:=1 to w do
        z:=h2(x);
end;
return z;
```

3.

Γίνεται χρήση του προγράμματος της άσκησης 2 (if … then … else).

1 Σε όλες τις λύσεις, με την πράξη της αφαίρεσης «-», εννοείται η διμελής πράξη που είναι κλειστή στους φυσικούς αριθμούς, καθώς δεν μπορεί να έχει αρνητικό αποτέλεσμα (monus). Αν αφαιρεθεί αριθμός από μικρότερό του, τότε το αποτέλεσμα είναι μηδενικό.

2 Το πρόγραμμα έχει έξοδο 0 αν $x=0$, και 1 αν $x \neq 0$.

***div*(*x*, *y*)** [3]

```
if y=0 then
        d:=0;
else
        for i:=1 to x do
                //d=mult(y, i+1)
                z:=0;
                for j:=1 to y do
                        for k:=1 to i+1 do
                                z:=z+1;
                        end;
                end;
                d:=z;
                //z=sub(d, x)-1, η διαφορά
                z:=x;
                for j:=1 to x+1 do
                        z:=z-1;
                end;
                if z=0 then
                        x:=0;
                        d:=i;
                else
                end;
        end;
end;
return d;
```

4.

***exp*(*x*, *y*)**

```
w:=1;
for i:=1 to y do
        //mult(w, x)
        z:=0;
        for j:=1 to x do
                for k:=1 to w do
                        z:=z+1;
                end;
        end;
        w:=z;
end;
return w;
```

3 Υπολογίζει το ακέραιο μέρος της διαίρεσης του y με το x.

***fact*(*n*)**

```
x:=1;
for i:=1 to n do
        //mult(x, i)
        z:=0;
        for j:=1 to i do
                for k:=1 to x do
                        z:=z+1;
                end;
        end;
        x:=z;
end;
return x;
```

***per*(*m*, *n*)**

```
//d=sub(m,n)
a:=m;
for i:=1 to n do
        a:=a-1;
end;
d:=a;
//pd=fact(d)
z:=0;
pd:=1;
for i:=1 to d do
        z:=0
        for j:=1 to i do
                for k:=1 to x do
                        z:=z+1;
                end;
        end;
        pd:=z;
end;
//pn=fact(n)
pn:=1;
z:=0;
for i:=1 to n do
        z:=0
        for j:=1 to i do
                for k:=1 to x do
                        z:=z+1;
                end;
        end;
        pn:=z;
end;
//pm=fact(m)
pm:=1;
z:=0;
for i:=1 to m do
        z:=0
        for j:=1 to i do
                for k:=1 to x do
                        z:=z+1;
                end;
        end;
        pm:=z;
end;
//s=mult(pn, pd)
s:=0;
for i:=1 to pd do
        for j:=1 to pn do
                s:=s+1;
        end;
end;
//r=div(pm, s)
if s=0 then
        r:=0;
else
        for i:=1 to pm do
                z:=0;
                for j:=1 to s do
                        for k:=1 to i+1 do
                                z:=z+1;
                        end;
                end;
                r:=z;
                z:=pm;
                for j:=1 to pm+1 do
                        z:=z-1;
                end;
                if z=0 then
                        pm:=0;
                        r:=i;
                else
                end;
        end;
end;
return r;
```

***isqrt*(*n*)**

```
for c:=1 to n do
        m:=n;
        //div(m,c)
        for i:=1 to m do
```

```
            z:=0;
            for j:=1 to c do
                for k:=1 to c+1 do
                    z:=z+1;
                end;
            end;
            d:=z;
            z:=m;
            for j:=1 to m+1 do
                z:=z-1;
            end;
            if z=0 then
                m:=0;
                d:=c;
            else
            end;
        end;
        if d=i then
            n=0;
            r:=d;
        else
            if d=i-1 then
                n=0;
                r:=d+1;
            else
            end;
        end;
end;
return r;
```

5.

i) ***divisible*** (m, n)

//Το πρόγραμμα είναι το div(m,n), αλλά ελαφρώς τροποποιημένο, ώστε υπολογίζοντας το υπόλοιπο της διαίρεσης να αποφαίνεται αν πρόκειται για τέλεια διαίρεση.

```
if n=0 then
        d:=0;
else
        for i:=1 to m do
            z:=0;
            for j:=1 to n do
                for k:=1 to i+1 do
                    z:=z+1;
                end;
            end;
            z:=m;
            for j:=1 to m+1 do
                z:=z-1;
            end;
            if z=n-1 then
                m:=0;
                d:=1;
            else
            end;
        end;
end;
return d;
```

ii) ***prime*** (n)

//Το πρόγραμμα εκτελεί επαναληπτικά το divisible(n, c+1) για c=1...n-2 και αν κάποιο c διαιρεί τέλεια τον n, επιστρέφει 0. Αλλιώς, 1.

```
d=1;
for c:=1 to n-2 do
        for i:=1 to n do
            z:=0;
            for j:=1 to c+1 do
                for k:= 1 to i+1 do
                    z:=z+1;
                end;
            end;
            z:=m;
            for j:=1 to n+1 do
                z:=z-1;
            end;
            if z=n-1 then
                n=0;
                d:=0;
            else
            end;
        end;
end;
return d;
```

iii) Για την κατασκευή αυτού του προγράμματος, κατασκευάζεται πρώτα ένα βοηθητικό πρόγραμμα, nextpr(n), που υπολογίζει τον ελάχιστο πρώτο αριθμό που είναι μεγαλύτερος του n. Έπειτα, ο κ-οστός πρώτος αριθμός υπολογίζεται κάνοντας κ επαναλήψεις του nextpr() ξεκινώντας με βάση το 2 και προχωρώντας κατά βήμα στον επόμενο πρώτο αριθμό.