ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

http://courses.softlab.ntua.gr/progintro/

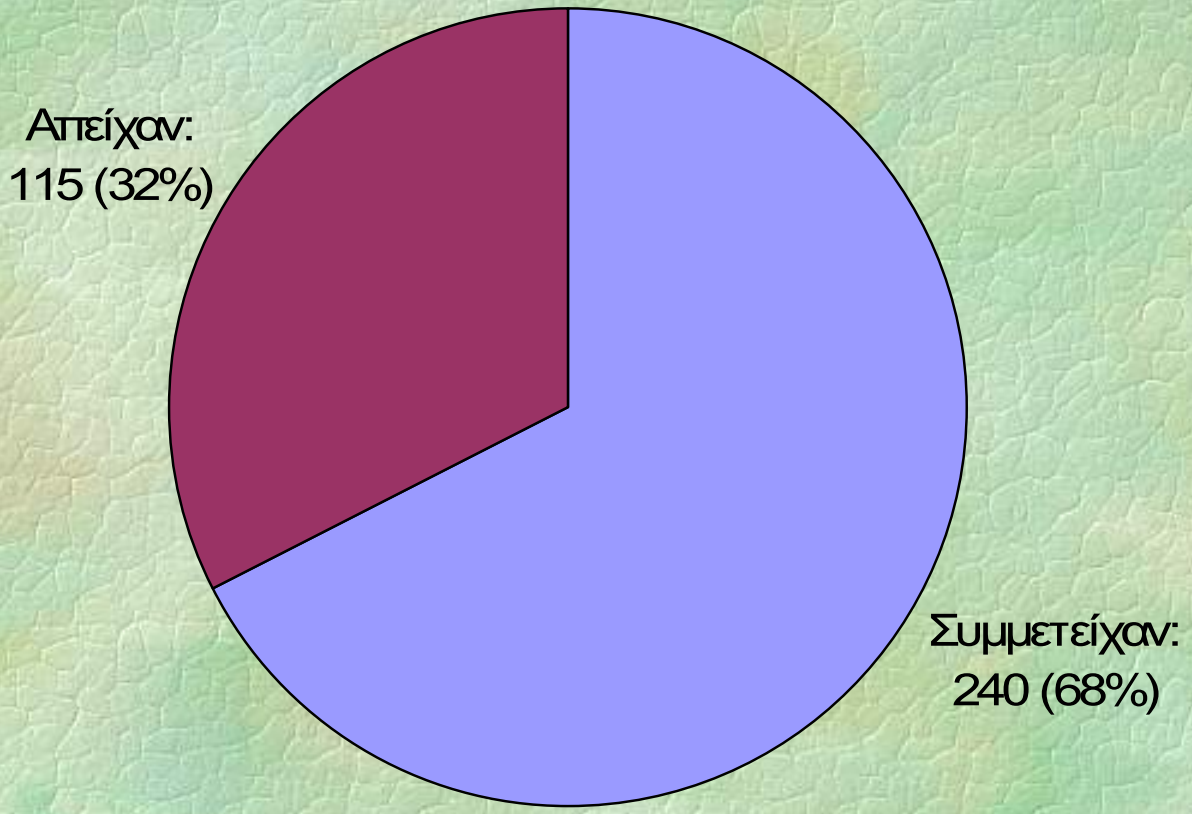
Διδάσκοντες: Στάθης Ζάχος (zachos@cs.ntua.gr)
Νίκος Παπασπύρου (nickie@softlab.ntua.gr)
Δημήτρης Φωτάκης (fotakis@cs.ntua.gr)

Αποτελέσματα προόδου

- Για να δούμε πώς τα πήγαμε...
- Στατιστικά βαθμολογίας
- Λύσεις θεμάτων
- Πίνακας βαθμολογίας στη σελίδα του μαθήματος

20/1/12

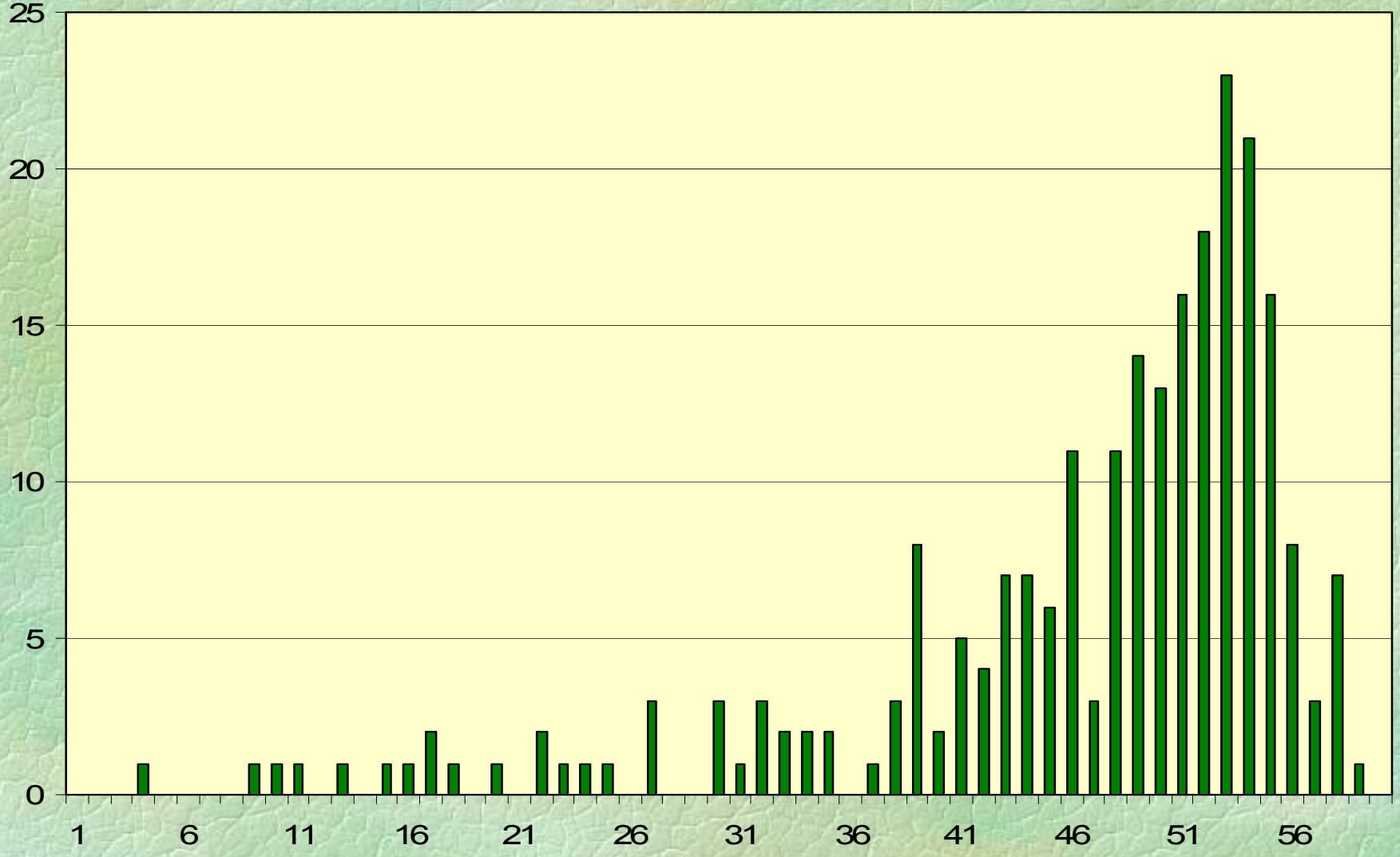
Συμμετοχή στην πρόοδο



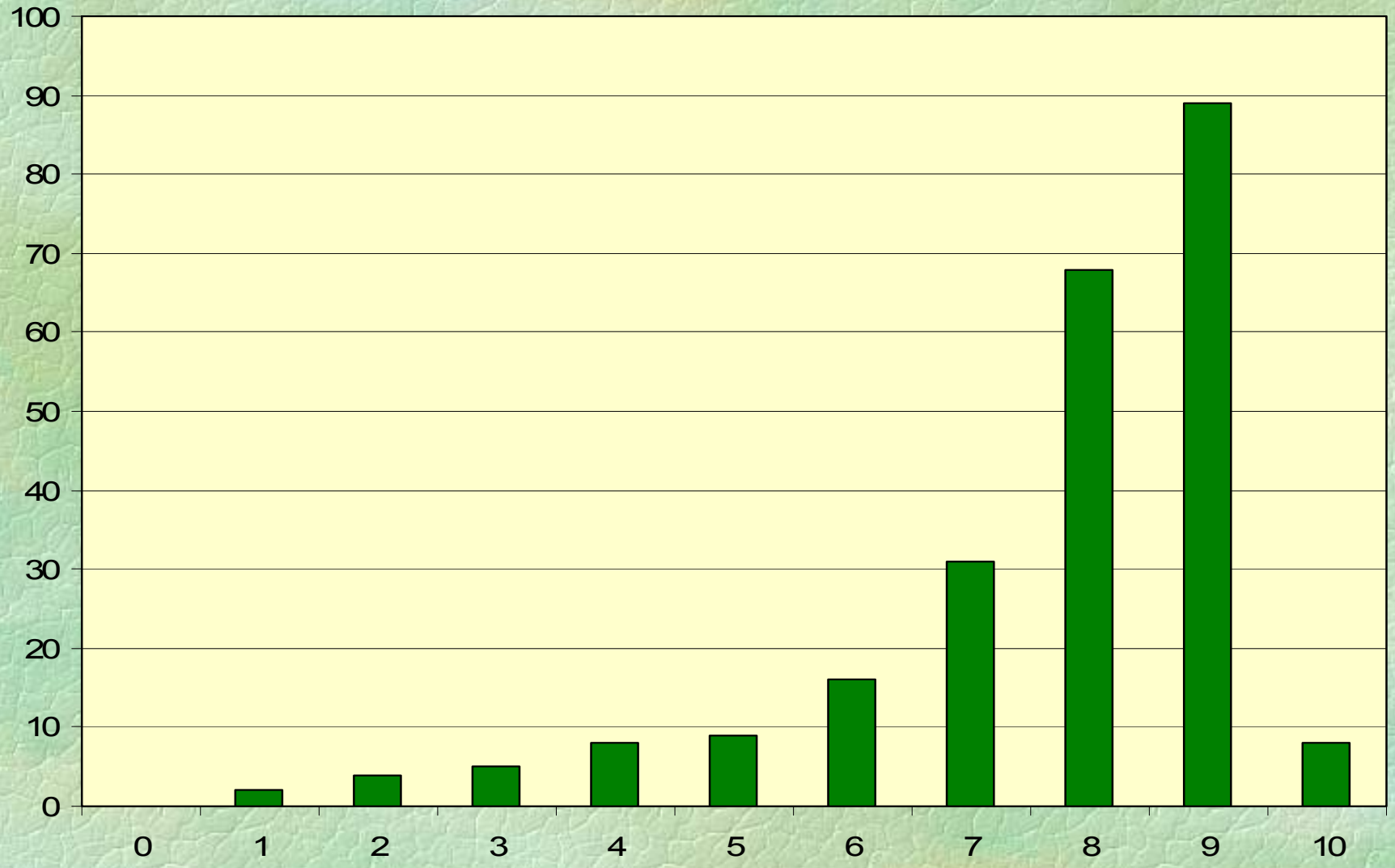
Βαθμολογία

Άριστα	60
Μέγιστο	59
Ελάχιστο	4
Μέσος όρος	46,396
Διάμεσος	50
Τυπική απόκλιση	10,502

Βαθμολογία, απόλυτη



Βαθμολογία, ανηγμένη στο 10

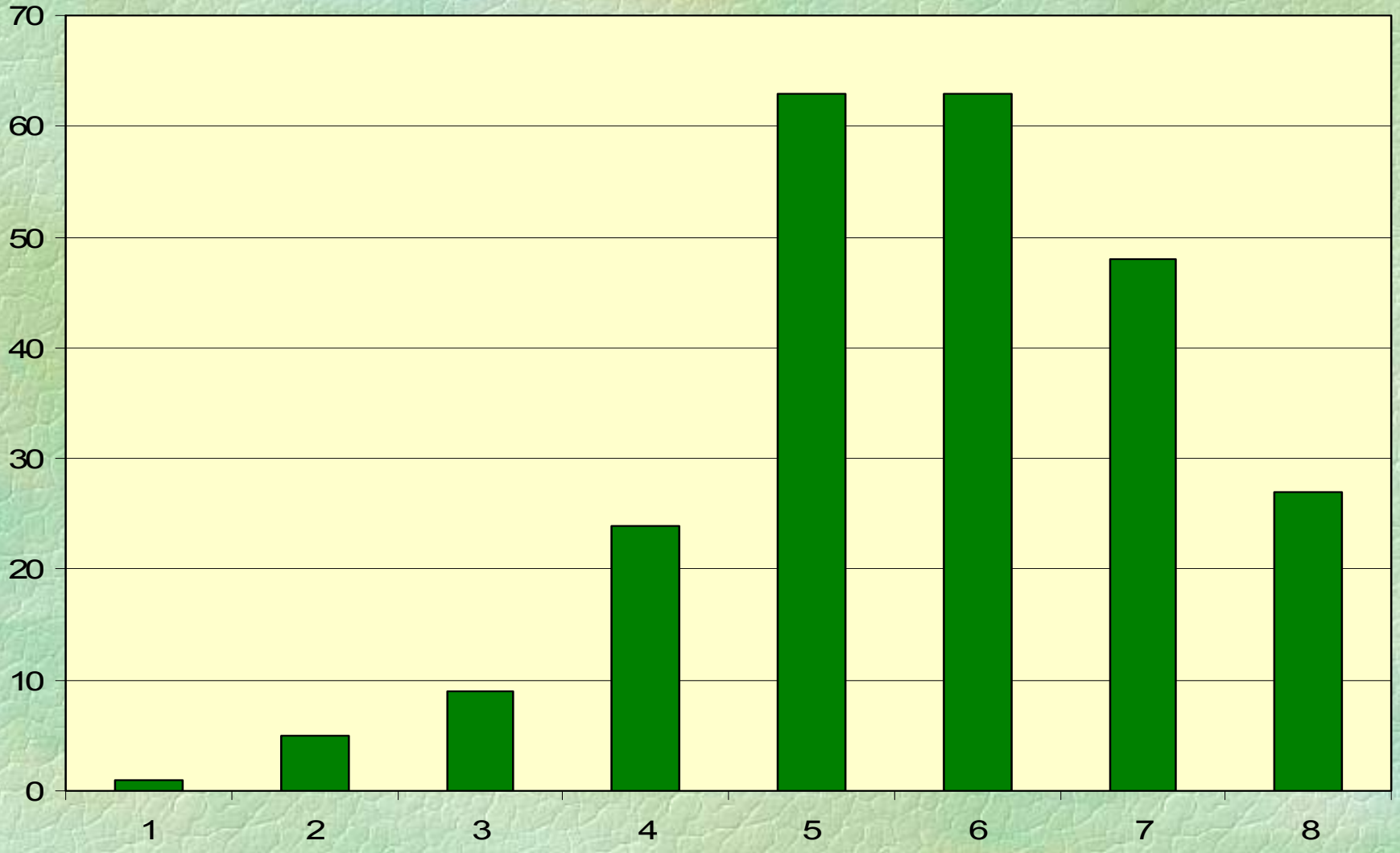


Θέμα 1

1. (8) Σημαδέψτε με κύκλο όλα τα λάθη που θα βρει ο μεταφραστής (compiler) στο παρακάτω πρόγραμμα. Αριθμήστε τους κύκλους και στη συνέχεια γράψτε τις αντίστοιχες τροποποιήσεις που προτείνετε για να διορθώσετε πρόγραμμα ώστε να μη διαμαρτύρεται ο compiler.

```
program p(input, output);
war   z     : real;
begin a, b : integer;
   readln(a, z);
   b := 42;
   whiLE (b >= 0) do
      writeln(a:3, z:8:4, b:5:3);
      if z >= 1.0 then z := z div 2;
                  else z := z * 1.5; b := B - 10; a := 2a;
   end
   writeln(round(z))
end
```

Θέμα 1, βαθμολογία

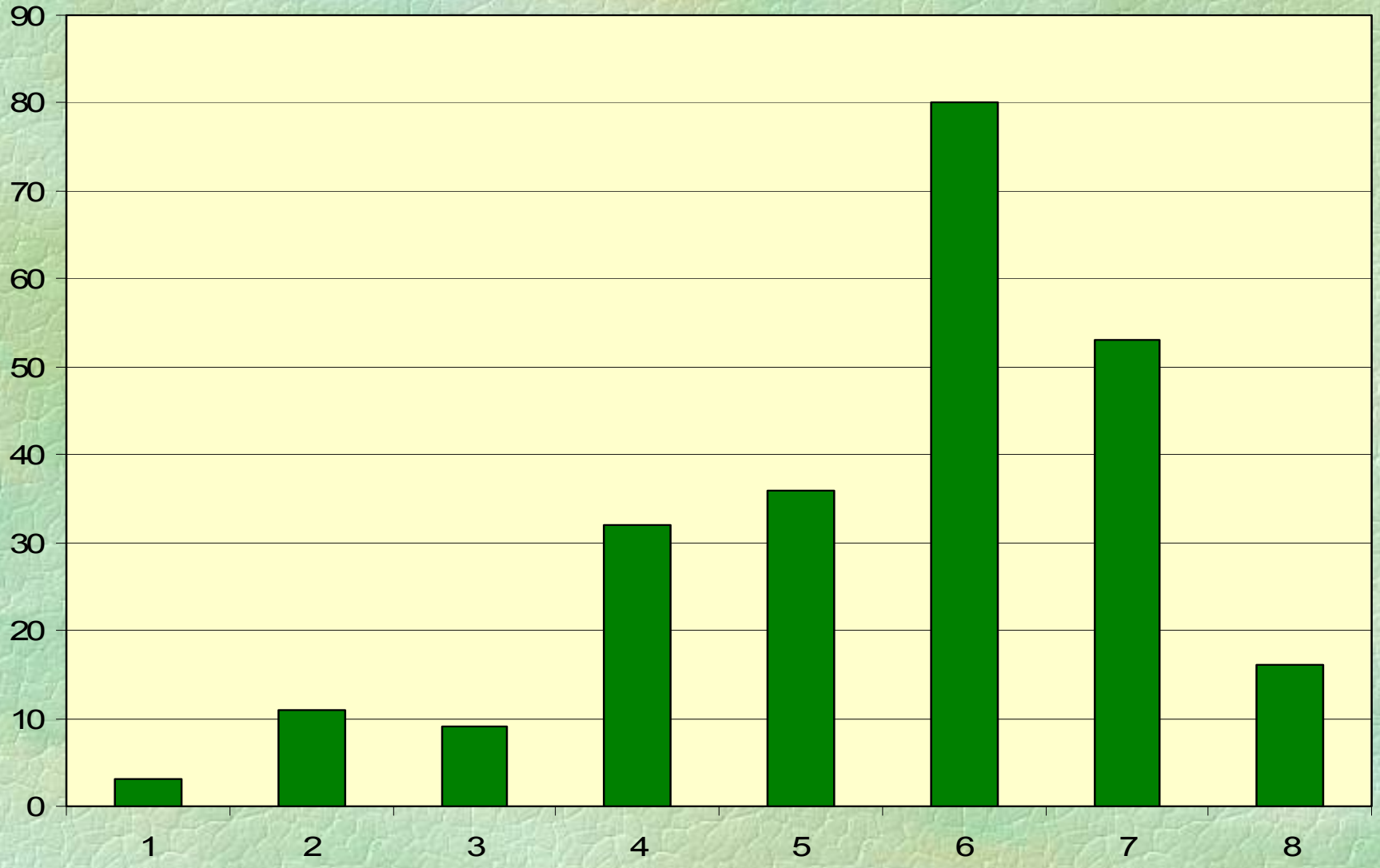


Θέμα 2

2. (8) Σημειώστε με κύκλο τις ορθές συντακτικά εντολές (statements) της Pascal.

1. `y = sqrt(x-12)`
2. `begin b[b[j]]:=b[j+1] end`
3. `repeat a:=a; b:=2*b; until a`
4. `ch := list[i] or (a<b)`
5. `if p then p:=a<b else k:=a-b`
6. `if a <= 6 then b:=c; c:=0 else b:=b+2`
7. `while arcta > 0 do`
8. `a+3 := b + i div 10`

Θέμα 2, βαθμολογία

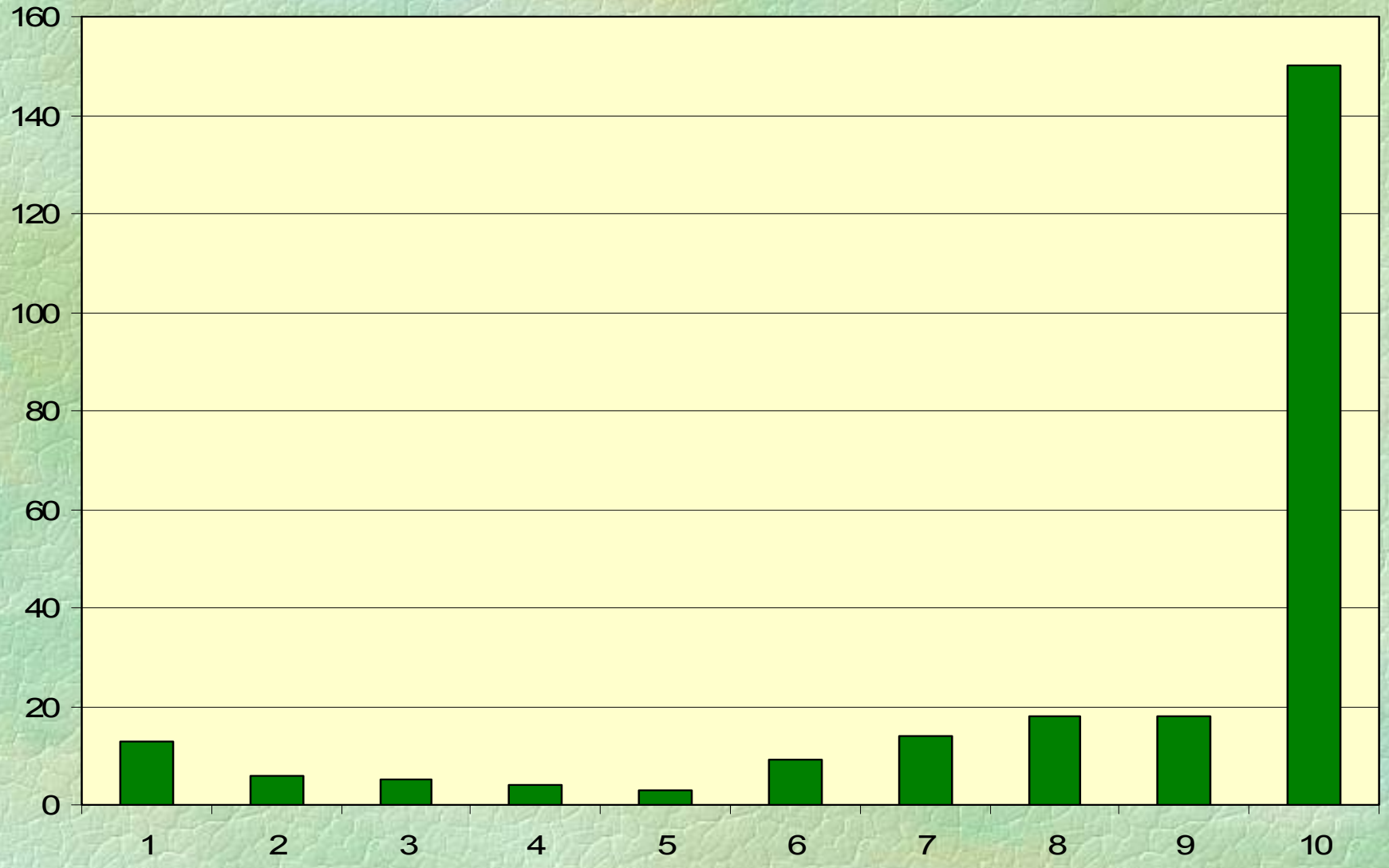


Θέμα 3

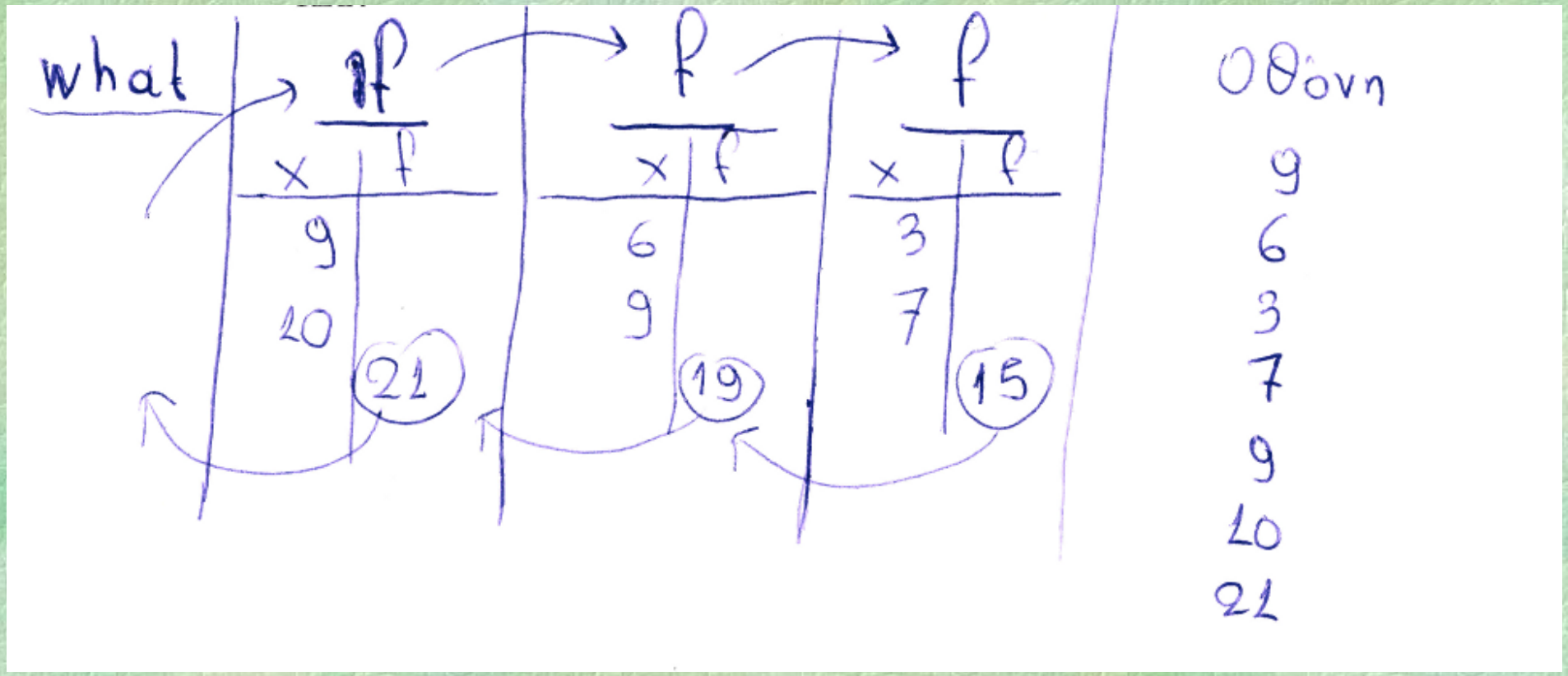
3. (10) Εκτελέστε με το χέρι. Δείξτε σε πίνακα όλες τις ενδιάμεσες τιμές καθώς και τις τιμές που εκτυπώνονται:

```
program what (output);

   function f (x : integer) : integer;
   begin writeln(x);
         if x > 4 then x := f(x-3) - x else x := 7;
         writeln(x);
         f := 2*x+1
   end;

begin writeln(f(9))
end.
```

Θέμα 3, βαθμολογία



Θέμα 3



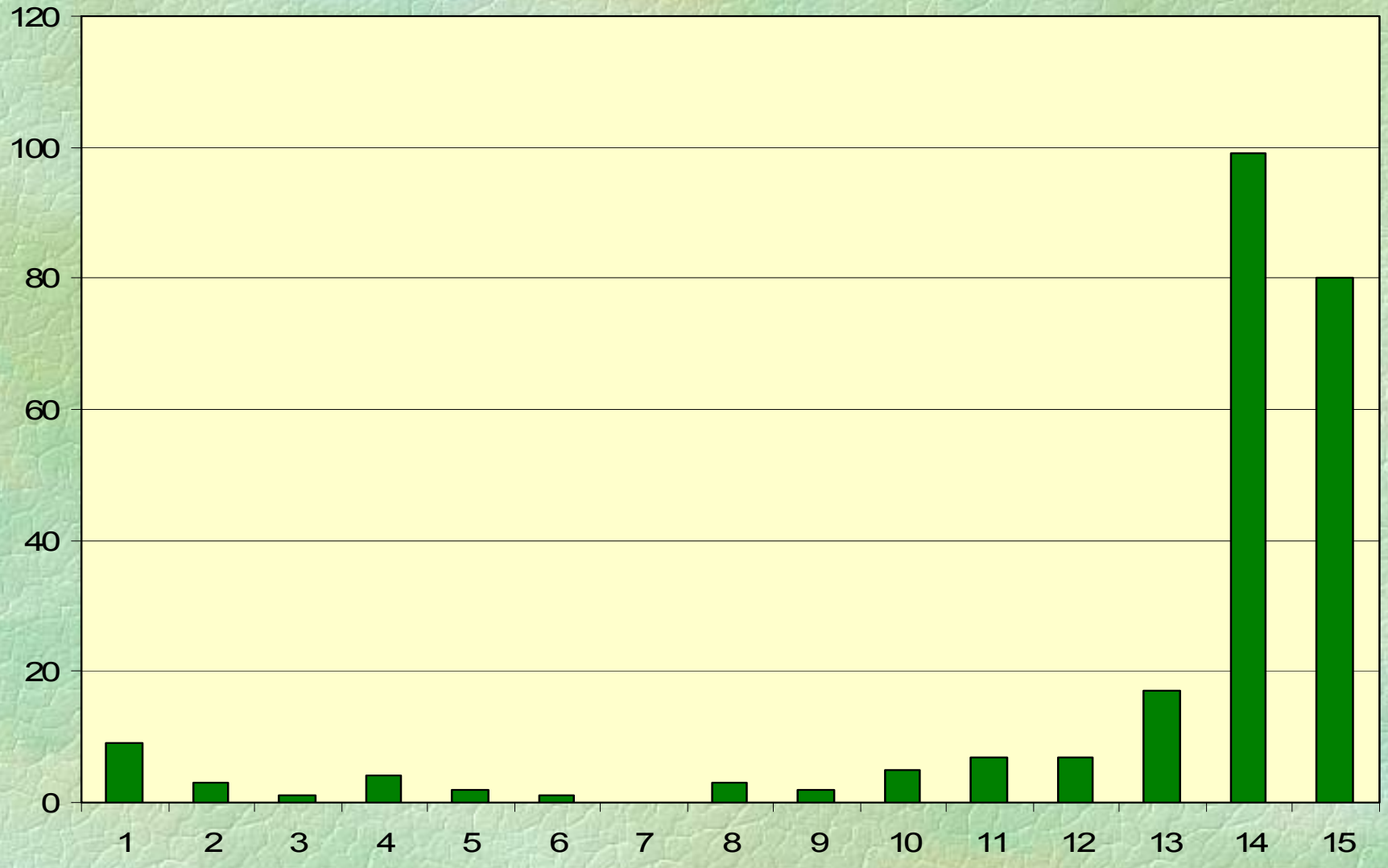
Θέμα 4

4. (15) Συμπληρώστε το επόμενο πρόγραμμα έτσι ώστε να διαβάζει μία συμβολοσειρά σε ένα πίνακα (array) **a** και να εκτυπώνει μία παλινδρομική συμβολοσειρά διπλάσιου μήκους, οι πρώτοι χαρακτήρες της οποίας είναι αυτοί που διαβάστηκαν. Υποθέστε ότι η αρχική συμβολοσειρά δεν υπερβαίνει τους 80 χαρακτήρες και ότι το τέλος της σηματοδοτείται από το τέλος γραμμής (eoln).

Για παράδειγμα, αν δοθεί η συμβολοσειρά "`neverod`", το πρόγραμμά σας θα πρέπει να εκτυπώνει τη συμβολοσειρά "`neveroddoreven`".

```
program palindrom (input, output);

   const max = 80;
```

Θέμα 4, βαθμολογία



Θέμα 4

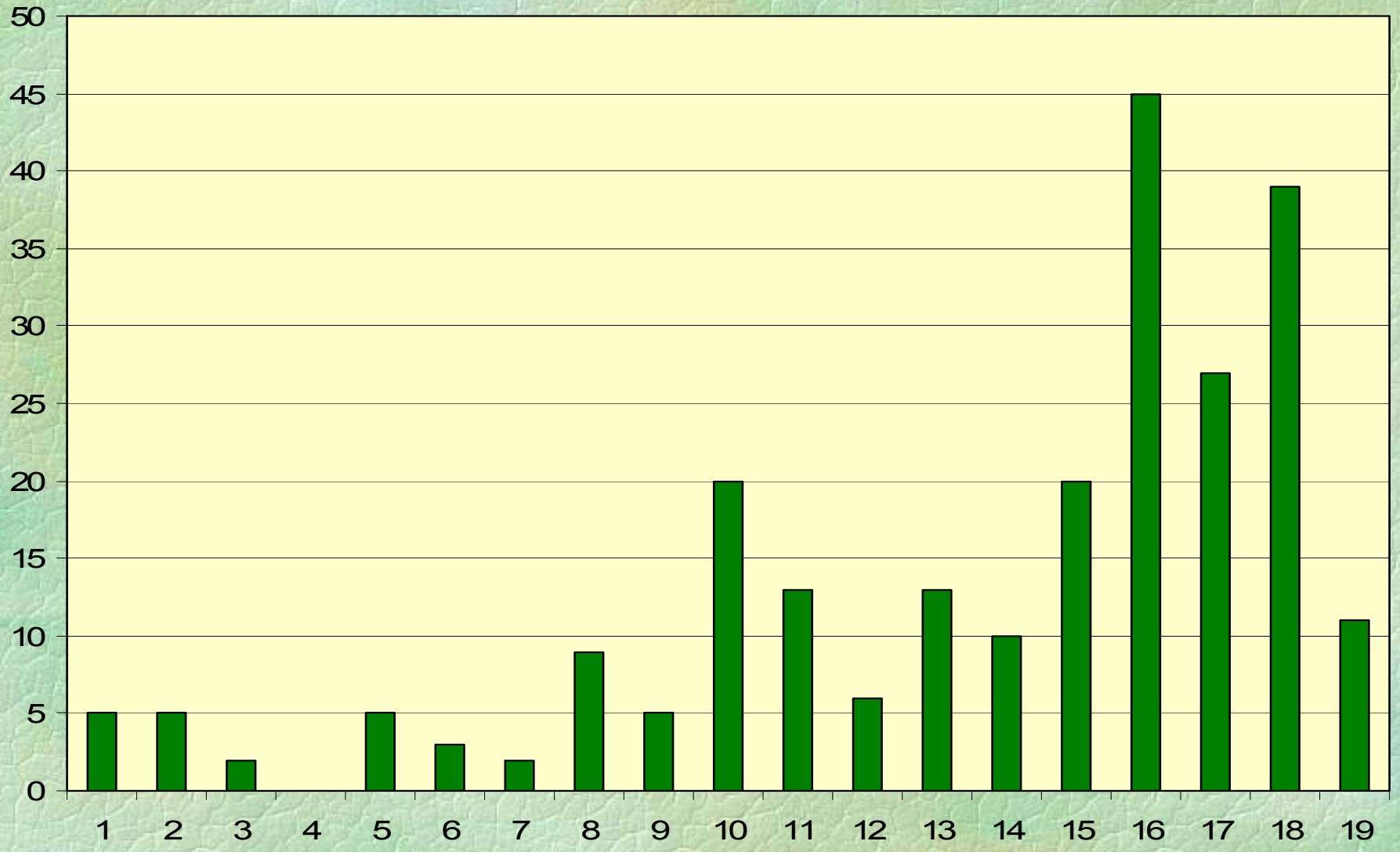
```
program palindrom (input, output);

   const max = 80;

   var i,j: integer;
       a: array[1..max] of char;

begin
   i:=0;
   while not eoln do begin
         i:=i+1;
         read(a[i])
   end;
   for j:=1 to i do write(a[j]);
   for j=i downto 1 do write(a[j])
end.
```

Θέμα 5

5. (19) Συμπληρώστε την ακόλουθη συνάρτηση έτσι ώστε να δέχεται ως παράμετρο ένα διδιάστατο πίνακα (array) **a** ακέραιων αριθμών με 10 γραμμές και 20 στήλες και να ελέγχει αν όλα τα στοιχεία της πρώτης γραμμής είναι περιττοί αριθμοί, όλα τα στοιχεία της δεύτερης είναι άρτιοι, όλα της τρίτης περιττοί, κ.ο.κ. Η συνάρτησή σας πρέπει να επιστρέφει `true` αν όλα τα στοιχεία κάθε γραμμής έχουν την παραπάνω ιδιότητα, διαφορετικά να επιστρέφει `false`. Δώστε προσοχή ώστε η συνάρτησή σας να είναι αποδοτική (π.χ. να μην κάνει άχρηστους υπολογισμούς).

```
const n = 10;  m = 20;
type  matrix = array [1..n, 1..m] of integer;
function checkAllLines (
```

Θέμα 5, βαθμολογία



Θέμα 5

```
const n = 10; m = 20;

type matrix = array [1..n, 1..m] of integer;

function checkAllLines (var a : matrix) : boolean;
  var i, j : integer;
      odd_this, all_good : boolean;
begin
  i := 0;  all_good := false;
  repeat  i := i+1;  odd_i := odd(i);
    j := 0;
    repeat  j := j+1;
      all_good := odd(a[i, j]) = odd_i
    until not all_good or (j >= m)
  until not all_good or (i >= m);
  checkAllLines := all_good
end
```

Θέμα 5

- Συνηθισμένα λάθη
  - αν δε χρησιμοποιούμε σταθερές
  - αν χρησιμοποιούμε for loop, δηλαδή δε σταματάμε μόλις βρούμε κάποιο στοιχείο που είναι εκτός θέσης
  - αν φτιάχνουμε και χρησιμοποιούμε περιττούς πίνακες
  - αν ελέγχουμε μία (πιθανώς σχετική) ιδιότητα αντί αυτού που θέλουμε πραγματικά να ελέγξουμε

Θέμα 5

Χρησιμοποιούμε τις σταθερές!

gram <= n

sthl <= m

```
const n = 10;   m = 20;
type  matrix = array [1..n, 1..m] of integer;
function checkAllLines ( a:matrix;) :boolean;
var   gram,sthl :integer;
      swsto : boolean ;

begin
  gram:=1;
  swsto:=true;
  while (gram<=10) and (swsto=true) do
    begin
    sthl:=1;
    while  (sthl<=20) and (swsto=true) do
       if  (gram mod 2 = 0) then
         if  (a[gram,sthl] mod 2 = 1) then
             swsto:=false
         else
             sthl:=sthl+1
       else
         if  (a[gram,sthl] mod 2 = 0) then
             swsto:=false
         else
             sthl:=sthl+1 .
    gram:=gram+1;
    end;
 checkAllLines:=swsto;
end;
```

ΠΡΟΣΟΧΗ:
όχι άριστη λύση!

Θέμα 5

Η χρήση for loop εδώ είναι κακή. Μπορούμε να σταματήσουμε νωρίτερα!

```
const n = 10;   m = 20;
type  matrix = array [1..n, 1..m] of integer;
function checkAllLines (a:matrix):boolean;
var   i,j : integer;
      flag : boolean;
  begin
     flag := true;
     for i:=1 to n do (*Με την εντολή while...do θα γλιτώναμε*)
        if flag then (*τις άσκοπες επαν. της for κατά τις οποίες δεν εκτε*)
           for j:=1 to m do (*λείται καμία εντολή όταν flag=false *)
              if flag then
                 if odd(i) then
                    flag:= odd(a[i,j])
                         else  flag:= not odd (a[i,j]);

     checkAllLines:= flag;

  end;
```

+17

ΠΡΟΣΟΧΗ: όχι καλή λύση!

Θέμα 5

Τρεις περιττοί πίνακες:

test[i]: true αν ο αριθμός i είναι ζυγός

counter1[j]: πλήθος των στοιχείων της γραμμής j που είναι ζυγά

counter2[j]: πλήθος των στοιχείων της γραμμής j που είναι μονά

Δε φτιάχνουμε και χρησιμοποιούμε πίνακες χωρίς λόγο!

```
const n = 10;  m = 20;
type  matrix = array [1..n, 1..m] of integer;
function checkAllLines (a:matrix):boolean;
 var i,j,k,sign1,sign2:integer;
     f_test:boolean;
     counter1, counter2: array [1..10] of integer;
      test: array [1..20] of boolean;
begin
  for i:=1 to m do if ((i mod 2)=0) then test[i]:=true
                   else test[i]:=false;
  sign1:=0;
  sign2:=0;
  for j:=1 to m do begin counter1[j]:=0; counter2[j]:=0
                   end;
  for j:=1 to m do if (test[j]=true) then
                   begin
                     for k:=1 to n do
                     if ((a[k,j] mod 2)=0) then
                       counter1[j]:=counter1[j]+1;
                   end
                   else begin
                     for k:=1 to n do
                     if (a[k,j]mod 2)=1 then
                       counter2[j]:=counter2[j]+1
                   end;
for j:=1 to m do
begin
  if counter1[j]<n then sign1:=1;
  if counter2[j]<n then sign2:=1;
end;
if (sign1=0) and (sign2=0) then f_test:=true
 else f_test:=false;
 checkALLines:=f_test;

end;
```

αντίγραφα
n ↔ m

ΠΡΟΣΟΧΗ: όχι καλή λύση!

Θέμα 5

Δεν ελέγχουμε μία ιδιότητα που (μπορεί να) είναι αναγκαία αλλά σίγουρα δεν είναι ικανή!

Αν μια γραμμή περιέχει ζυγούς αριθμούς, το άθροισμά της είναι επίσης ζυγός.

Όχι όμως αντίστροφα!

Και όχι το ίδιο για μονούς!

```
const n = 10;   m = 20;
type  matrix = array [1..n, 1..m] of integer;
function checkAllLines (a:matrix): boolean;
     Var [illegible] i, j, sum : integ
         stop : boolean;
begin
   i:=1; stop:=false;
   repeat
      sum:=0;
      for j:=1 to m do
         sum:=sum+a[i,j];
      if odd(i) then if odd(sum) then stop:=false
                                 else stop:=true
                else if odd(sum) then stop:=true
                                 else stop:=false;
      i:=i+1
   until stop or (i>n);
   if stop then checkAllLines:=false
           else checkAllLines:=true
end;
```

ΠΡΟΣΟΧΗ: λάθος λύση!