Ε. Μ. Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. & Μηχ. Υπολογιστών.
Ε. Ζάχος, Ν. Παπασπύρου, Δ. Φωτάκης, Δ. Σούλιου, Π. Ποτίκας

ΕΠΩΝΥΜΟ:	
ΟΝΟΜΑ:	
ΑΡ. ΜΗΤΡΩΟΥ:	
ΕΞΑΜΗΝΟ:	
ΟΜΑΔΑ ΕΡΓ:	
ΑΜΦΙΘΕΑΤΡΟ:	
ΘΕΣΗ:	

1	0
2	8
3	0
4	7
5	6
6	10
ΣΥΝΟΛΟ	31

A

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ Η/Υ

Κανονική εξέταση, Φεβρουάριος 2017

= για παλιό είναι ≈ 7 ή 6
= για νέο εξαρτάται από εργαστήριο

Κανονισμός εξέτασης: *1) Υποχρεούστε να δείξετε στον επιτηρητή όταν σας ζητηθεί τη φοιτητική σας ταυτότητα ή άλλο αποδεικτικό της ταυτότητάς σας με φωτογραφία. 2) Η εξέταση γίνεται με κλειστά βιβλία και σημειώσεις. 3) Δεν μπορείτε να χρησιμοποιείτε ηλεκτρονικές συσκευές. Αν έχετε μαζί σας κινητό τηλέφωνο, απενεργοποιήστε το και κρύψτε το.*

1. (8)

Να δείξετε σε **πίνακα** όλες τις **ενδιάμεσες τιμές** καθώς και τις **τιμές που τυπώνονται** από το παρακάτω πρόγραμμα (εκτέλεση με το χέρι). Δεξιά, το ίδιο πρόγραμμα σε απλή C.

```
int x=2, y=4, z=17, w=3;

PROC p(int z, int &x) {
  x = 3*z - x;
  int w = x + z;
  WRITELN(x, y, z, w);
  if (x <= y) {
    p(x+1, w);
    WRITELN(x, y, z, w);
  }
}

PROGRAM f17a() {
  p(1, x);
  WRITELN(x, y, z, w);
}
```

```
#include <stdio.h>

int x=2, y=4, z=17, w=3;

void p(int z, int *x) {
  *x = 3*z - *x;
  int w = *x + z;
  printf("%d %d %d %d\n", *x, y, z, w);
  if (*x <= y) {  p(*x+1, &w);
    printf("%d %d %d %d\n", *x, y, z, w);
  }
}

int main() { p(1, &x);
  printf("%d %d %d %d\n", x, y, z, w);
  return 0; }
```

Βαριέμαι να κάνω πράξεις και σκοπεύω να περάσω χωρίς αυτό το θέμα.

8

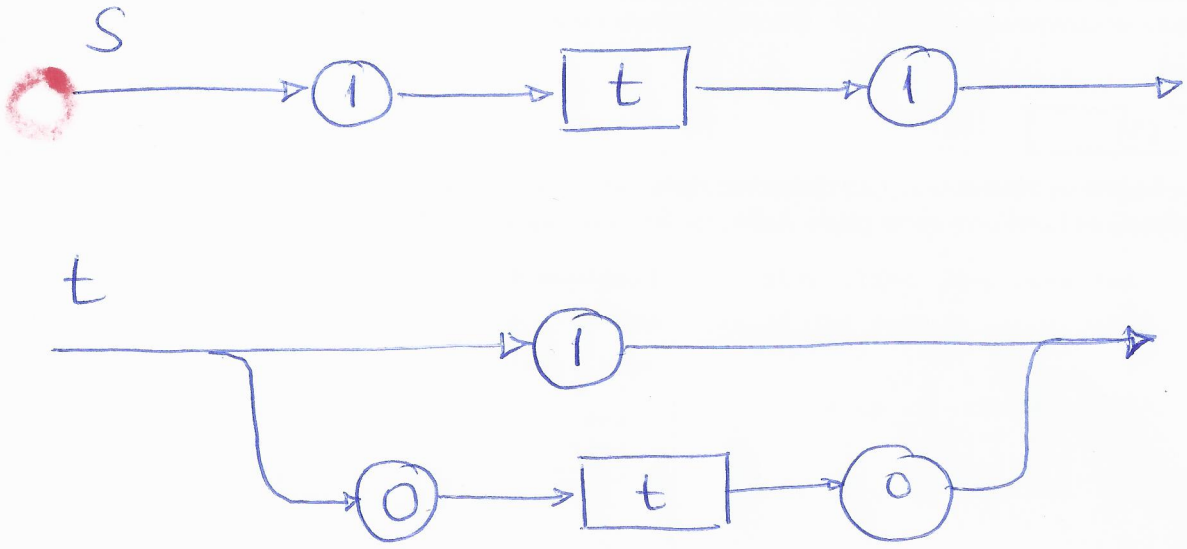
2. (8)

Κατασκευάστε **συντακτικό διάγραμμα** που περιγράφει τη δυαδική αναπαράσταση όλων των αριθμών της μορφής $2^n\,(2^n+1)+1$, όπου $n \geq 1$. Οι πέντε μικρότεροι τέτοιοι αριθμοί είναι οι εξής:

111 **10101** **1001001** **100010001** **10000100001** ...

Δηλαδή, το συντακτικό διάγραμμα πρέπει να περιγράφει τις συμβολοσειρές που:

- αποτελούνται από **0** και **1**,
- έχουν περιττό πλήθος ψηφίων, και
- έχουν ακριβώς τρία **1**: στην αρχή, στη μέση και στο τέλος.



0

βαριέμαι και δεν καθόλου αισθάνομαι τυχερός.

3. (10)

Απαντήστε στις παρακάτω ερωτήσεις πολλαπλής επιλογής μαυρίζοντας σε κάθε μία το πολύ ένα από τα τέσσερα κουτάκια. Κάθε σωστή απάντηση παίρνει 1,5 μονάδα. Κάθε λάθος απάντηση χάνει 0,5 μονάδα (αρνητική βαθμολογία). Κενές ή άκυρες απαντήσεις δεν προσθέτουν ούτε αφαιρούν μονάδες.

(α) Έστω ότι έχετε τρεις διαφορετικούς αλγορίθμους Α, Β και Γ, που επιλύουν το ίδιο πρόβλημα. Η πολυπλοκότητα του Α είναι $O(n \log^5 n)$, του Β είναι $O(n^2)$, και του Γ είναι $O(2^n)$. Ποιον από τους τρεις θα προτιμούσατε; (Θεωρήστε ότι μας ενδιαφέρουν μεγάλες τιμές του n.)

- ☐ τον Α
- ☐ τον Β
- ☐ τον Γ
- ☐ οποιονδήποτε από τους Α ή Β, δεν έχουν διαφορά

(β) Ο καλύτερος αλγόριθμος για την αναζήτηση στοιχείου σε ένα δυαδικό δέντρο αναζήτησης με n στοιχεία:

- ☐ απαιτεί χρόνο $O(1)$ στη χειρότερη περίπτωση.
- ☐ απαιτεί χρόνο $O(\log n)$ στη χειρότερη περίπτωση.
- ☐ απαιτεί χρόνο ανάλογο του ύψους του δέντρου στη χειρότερη περίπτωση.
- ☐ απαιτεί χρόνο ανάλογο του n στη χειρότερη περίπτωση.

(γ) Ποια είναι η τιμή της μεταβλητής **n** στο τέλος της εκτέλεσης του ακόλουθου τμήματος προγράμματος;

```
n=2048; p=1;
do { n=n/2; p=2*p; } while (n >= 8);
n=p+n;
```

☐ 144 ☐ 264 ☐ 516 ☐ 2048

(δ) Ποια είναι η τιμή της μεταβλητής **t** στο τέλος της εκτέλεσης του ακόλουθου τμήματος προγράμματος; Δεξιά το ίδιο πρόγραμμα σε απλή C.

```
n=1024; t=0; i=1;
while (i <= n) {
  n=n/2; i++;
  FOR (j, 1 TO n) t++; }
```

```
n=1024; t=0; i=1;
while (i <= n) {
  n=n/2; i++;
  for (j=1; j<=n; j++) t++; }
```

☐ 1048576 ☐ 1016 ☐ 1023 ☐ 1020

(ε) Ποιο από τα παρακάτω προγράμματα τυπώνει 42;

```
int k=3;

PROC proc1(int *n) {
  *n=k*(k+1)+2; WRITELN(*n*k);
}

PROGRAM test1() { proc1(&k); }
```

```
int k=3;

PROC proc2(int n) {
  n=k*(k+1)+2; WRITELN(n*k);
}

PROGRAM test2() { proc2(k); }
```

☐ test1 ☐ test2 ☐ Και τα δύο ☐ Κανένα από τα δύο

(ζ) Τι θα επιστρέψει η παρακάτω συνάρτηση αν κληθεί με **x = 125** και **y = 35**;

```
int fun(int x, int y){
    if (y <= 1) return x;
    else if (y > x / y) return fun(y, x / y);
         else return fun(x / y, y);   }
```

☐ 3 ☐ 5 ☐ 11 ☐ 35

(η) Τι τυπώνει το παρακάτω πρόγραμμα;

```
PROGRAM pointers() {
  int *p, *q, *t;
  p = new int; q = new int; t = new int;
  *p=17; *q=7; *t=42;
  p=t; t=q;
  *t=*p**q ; *q=*q+*t;
  WRITELN(*t);
}
```

☐ 17 ☐ 42 ☐ 98 ☐ 588

ΠΡΟΧΕΙΡΟ

5+2

4. (10)

Αν γράψετε μόνο τη λέξη «ΚΕΝΟ» αντί λύσης σε αυτό το θέμα, θα πάρετε 2 μονάδες.

Θεωρούμε έναν μονοδιάστατο πίνακα **A** με **N** ακέραιους αριθμούς. Το <u>γινόμενο στοιχείων-θέσεων</u> του **A** είναι ίσο με $\sum_{k=1}^{N} \mathbf{A}[k] \times k$, είναι δηλαδή το άθροισμα των γινομένων που σχηματίζονται όταν πολλαπλασιάσουμε κάθε στοιχείο του πίνακα με τη θέση του. Π.χ., το γινόμενο στοιχείων-θέσεων του πίνακα **A** = [5, 3, 2, 6, 4, 1] είναι $5 \cdot 1 + 3 \cdot 2 + 2 \cdot 3 + 6 \cdot 4 + 4 \cdot 5 + 1 \cdot 6 = 67$.

Η περιστροφή ενός πίνακα **A** κατά **p** θέσεις (όπου $0 \le \mathbf{p} < \mathbf{N}$) μετασχηματίζει τον **A** παίρνοντας τα **p** πρώτα στοιχεία του και βάζοντάς τα στο τέλος. Π.χ., η περιστροφή του **A** = [5, 3, 2, 6, 4, 1] κατά 2 θέσεις δίνει [2, 6, 4, 1, 5, 3].

Να γράψετε μία *<u>κομψή και αποδοτική</u>* συνάρτηση που δέχεται ως παραμέτρους τα **A** και **N**, και υπολογίζει την περιστροφή του πίνακα **A** με το <u>**ελάχιστο**</u> γινόμενο στοιχείων-θέσεων. Η συνάρτηση πρέπει να τυπώνει τα στοιχεία του πίνακα **A** μετά από αυτή την περιστροφή και να επιστρέφει το αντίστοιχο γινόμενο.

<u>Παράδειγμα 1</u>: (**N** = 6)

A = [5, 3, 2, 6, 4, 1]
min_elem_pos_prod(**A**, **N**) = 64
Οθόνη: 6 4 1 5 3 2

<u>Παράδειγμα 2</u>: (**N** = 8)

B = [3, 4, 8, 1, 2, 5, 7, 6]
min_elem_pos_prod(**B**, **N**) = 140
Οθόνη: 5 7 6 3 4 8 1 2

Για τις 6 περιστροφές του **A** = [5, 3, 2, 6, 4, 1] δίνονται τα αντίστοιχα γινόμενα στοιχείων-θέσεων:

για p = 0: 67	[5, 3, 2, 6, 4, 1]
για p = 1: 76	[3, 2, 6, 4, 1, 5]
για p = 2: 73	[2, 6, 4, 1, 5, 3]
για p = 3: **64**	[6, 4, 1, 5, 3, 2]
για p = 4: 79	[4, 1, 5, 3, 2, 6]
για p = 5: 82	[1, 5, 3, 2, 6, 4]

Ερώτηση bonus (2 επιπλέον μονάδες): Ποια είναι η πολυπλοκότητα της λύσης σας; Εξηγήστε.

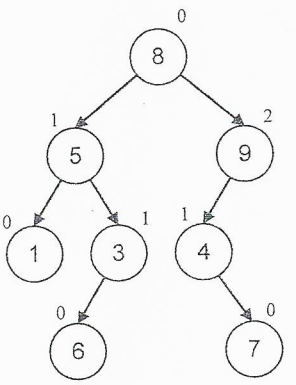
```
FUNC int min_elem_pos_prod ( int A[] , int N ) {
    int best_p = -1 , best ;
    int p, k ;
    FOR ( p , 0 TO N-1 ) {
        int sum = 0 ;
        FOR ( k, 0 TO N-1 )
            sum = sum + (k+1) * A[(k+p) % N] ;
        if ( best_p<0 || sum < best ) {
            best_p = p ;  best = sum ;
        }
    }
    FOR ( k, 0 TO N-1 ) {
        WRITE( A[(k+ best_p) % N]);
        if (k==N-1) WRITELN() ; else WRITE(" ");
    }
    return best ;
}
```

BONUS: προφανώς $O(N^2)$

6

5. (10)

Αν γράψετε μόνο τη λέξη «ΚΕΝΟ» αντί λύσης σε αυτό το θέμα, θα πάρετε 2 μονάδες.



Ορίστε τον τύπο **tree** του δυαδικού δέντρου που περιέχει στους κόμβους του ακέραιους αριθμούς, καθώς και τον τύπο **node** του κόμβου του.

Για κάθε κόμβο του δέντρου, ορίζουμε ως ***ανισορροπία*** την απόλυτη τιμή της διαφοράς των υψών του αριστερού και του δεξιού του παιδιού. Για παράδειγμα, στο δέντρο του διπλανού σχήματος φαίνονται οι τιμές της ανισορροπίας κάθε κόμβου.

Γράψτε μια *κομψή και αποδοτική* συνάρτηση που δέχεται ως παράμετρο ένα τέτοιο δέντρο και επιστρέφει τη ***μέγιστη*** τιμή της ανισορροπίας των κόμβων του. Η συνάρτηση θα πρέπει να έχει ως επικεφαλίδα:

```
FUNC int unbalance (node *t);
```

Για το παράδειγμα του διπλανού σχήματος, η συνάρτησή σας θα πρέπει να επιστρέφει 2. Σε ένα κενό δέντρο, η συνάρτησή σας πρέπει να επιστρέφει 0.

```
// από διαφάνειες, ό,τι θυμαμαι

struct node {
    int info;
    node * left, * right;
};
typedef node * tree;

// από διαφάνειες

FUNC int height ( tree t) {
    if (t == NULL) return 0;
    return 1 + max ( height ( t -> left) , height(t -> right));
}

// παραλλαγή του size, από διαφάνειες,
// σε O(N^2), ξέρω ότι δεν παίρνω άριστα.

FUNC int unbalance (tree t) {
    if (t == NULL) return 0;
    int ~~result = 0;~~
    ~~if (t == NULL) {~~
    int ~~height~~ this_node = abs ( height (t -> left)
                                   - height (t -> right));
    int unbalance_left = unbalance( t -> left);
    int unbalance_right = unbalance (t -> right);
    return max (this_node, max (unbalance_left,
                                unbalance_right));
}
```

10

6. (10)

Αν γράψετε μόνο τη λέξη «ΚΕΝΟ» αντί λύσης σε αυτό το θέμα, θα πάρετε 2 μονάδες.

Ζητείται ένα *κομψό και αποδοτικό* πρόγραμμα που διαβάζει από το αρχείο με όνομα "file.txt" ένα κείμενο. Το πρόγραμμά σας πρέπει να εκτυπώνει στην οθόνη όλους τους αριθμούς που εμφανίζονται μέσα σε αυτό, έναν σε κάθε γραμμή. Ως αριθμός εννοείται οποιαδήποτε συμβολοσειρά αποτελείται από ένα ή περισσότερα δεκαδικά ψηφία.

Παράδειγμα:

(κείμενο):	(οθόνη):
Paul Raymond Hudak (born July 15th, 1952, died April 29th, 2015) was an American professor of computer science at Yale University who was best known for his involvement in the design of the Haskell programming language, as well as several textbooks on Haskell and computer music. He was co-author of one of 50 papers chosen for inclusion in "Twenty Years of PLDI (1979-1999): A Selection".	15 1952 29 2015 50 1979 1999

```
PROGRAM   textproc ()  {
    INPUT ("file.txt");
    int  c = getchar();
    while ( c != EOF ) {
        if (isdigit(c)) {
            do { putchar(c);
                 c= getchar(); } while (isdigit(c));
            putchar('\n');
        }
        else  c = getchar();
}

// Η isdigit υπάρχει στο #include <ctype>
// αλλά είναι αυτή:

FUNC bool isdigit (char c) {
    return  c >= '0' && c <= '9';
}
```