Κανονική εξέταση, Ιανουάριος 2025

Σύνολο Μονάδων 60 – Διάρκεια Εξέτασης: 2:15

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 επιτρέπεται να χρησιμοποιείτε κινητό τηλέφωνο, αριθμομηχανή ή άλλη ηλεκτρονική συσκευή. Σε περίπτωση που έχετε μαζί σας κάτι από τα παραπάνω, παρακαλούμε, απενεργοποιήστε τα και κρύψτε τα. 4) Η αποχώρηση είναι δυνατή μετά την παρέλευση τουλάχιστον 60' από την έναρξη της εξέτασης.*

1. (10)

Να δείξετε σε **πίνακα** όλες τις **ενδιάμεσες τιμές των μεταβλητών** καθώς και τις **τιμές που τυπώνονται** από το παρακάτω πρόγραμμα C++ (εκτέλεση με το χέρι).

```
#include "pzhelp"
int a = 6, b = 8, c = 12;
void p(int a, int &c) {
  int b = c + a++;
  WRITELN(a, b, c);
  if (a > c) { p(a, b); WRITELN(a, b, c); }
  else { a *= 2; WRITELN(a, b, c); }
}
OGRAM { p(b, a); WRITELN(a, b, c); }
```

2. (14)

*Απαντήστε στις παρακάτω ερωτήσεις πολλαπλής επιλογής μαυρίζοντας σε κάθε μία το πολύ ένα από τα τέσσερα κουτάκια. Κάθε **σωστή** απάντηση **παίρνει 2 μονάδες**. Κάθε **λάθος** απάντηση **χάνει 0,7 μονάδες** (αρνητική βαθμολογία). Κενές ή άκυρες απαντήσεις δεν προσθέτουν ούτε αφαιρούν μονάδες.*

(α) Τι θα επιστρέψει η κλήση `f(5, 7);`

```
int f(int x, int c) {
  x /= 2;
  if (x < 1) return 1;
  c = 2 * c;
  return c + f(x, c);
}
```

☐ 1 ☐ 29 ☐ 42 ☐ κανένα από αυτά

(β) Ποια από τις ακόλουθες αναλλοίωτες για το σημείο 1 είναι σωστή και επαρκής, ώστε να αποδείξουμε ότι η βεβαίωση res = n! ισχύει στο σημείο 2;

```
int res = 1;
for (int i = 1; i <= n; ++i) { /*1*/ res *= i; }
/*2*/
```

☐ res = (i-1)! ☐ res = i! και i <= n ☐ res = (i-1)! και i <= n ☐ καμία από αυτές

(γ) Ποια είναι η πολυπλοκότητα χρόνου του παρακάτω κώδικα (ως συνάρτηση των N και M);

```
int a = 0, b = 0;
for (int i = 0; i < N; i++) {
  b += i%3; j = 1;
  while (j < M) { j *= 2; a += j%2; }
}
```

☐ O(N*logM) ☐ O(N+M) ☐ O(M+logN) ☐ O(N+logM)

(δ) Τι θα επιστρέψει η κλήση f(11, 27);

```
int f(int m, int k) {
  int count = 0;
  for (int i = 1; i <= 1000; ++i) {
    if (i % m == 0) continue; else if (i % k == 0) break;
    ++count;
  }
  return count;
}
```

☐ 11 ☐ 24 ☐ 27 ☐ κανένα από αυτά

(ε) Καλείται η συνάρτηση solve(4, 1, 2, 3). Τι τυπώνεται στην 3η γραμμή εκτύπωσης;

```
void solve(int rings, int source, int target, int auxil) {
  if (rings == 0) return;
  solve(rings-1, source, auxil, target);
  WRITELN("from", source, "to", target);
  solve(rings-1, auxil, target, source);
}
```

☐ from 1 to 3 ☐ from 1 to 2 ☐ from 3 to 2 ☐ κανένα από αυτά

(στ) Τι θα τυπωθεί στην οθόνη από τον παρακάτω κώδικα;

```
int a[] = {32, 26, 50, 1, 8, 9, 42, 15, 17, 20};
int *p = &a[4], *q = &a[6];
p = q-- ;
WRITELN(*p, *q, a[3]);
```

□ 42 15 1 □ 42 9 1 □ 8 9 1 □ 42 9 50

(ζ) Τι πολυπλοκότητα έχει η συνάρτηση υπολογισμού του ύψους ενός δυαδικού δέντρου, αν N είναι το πλήθος των κόμβων του δέντρου και K το ύψος του;

□ O(N) □ O(logN) □ O(K) □ κανένα από αυτά

ΠΡΟΧΕΙΡΟ

3. (12)

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

Ζητείται ένα <u>κομψό και αποδοτικό</u> πρόγραμμα που διαβάζει από ένα αρχείο με όνομα **file.txt** κείμενο αποτελούμενο από πεζά λατινικά γράμματα, κενά διαστήματα και αλλαγές γραμμής. Το πρόγραμμά σας πρέπει να εκτυπώνει στην οθόνη το πλήθος των γραμμών που έχουν τουλάχιστον δύο λέξεις και που η πρώτη λέξη είναι η ίδια με την τελευταία. Ως λέξη εννοείται οποιαδήποτε συμβολοσειρά αποτελείται από γράμματα και χωρίζεται με κενά διαστήματα (ή την αρχή ή το τέλος γραμμής) από το υπόλοιπο κείμενο.

Για το 70% της βαθμολογίας, μπορείτε να θεωρήσετε ότι οι λέξεις χωρίζονται μεταξύ τους με ένα μόνο κενό και ότι δεν υπάρχουν κενά στην αρχή και στο τέλος των γραμμών.

Προσοχή: Μη χρησιμοποιήσετε την κλάση std::string από τη βιβλιοθήκη STL της C++ και εν γένει έτοιμες κλάσεις και συναρτήσεις βιβλιοθήκης για συμβολοσειρές.

Παράδειγμα:

(αρχικό κείμενο):

```
only
here two
three here three

two two
four one two four
  five five five five five
 six  six  six     six  not
```

(οθόνη):

```
4
```

4. (12)

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

Ορίστε τον τύπο **node** των κόμβων γραμμικής λίστας που περιέχουν ως πληροφορία έναν ακέραιο αριθμό.

Γράψτε μια <u>*κομψή και αποδοτική*</u> συνάρτηση που δέχεται ως παράμετρο μια γραμμική λίστα **p**. Η συνάρτηση θα πρέπει να ταξινομεί τα περιεχόμενα των κόμβων της λίστας σε αύξουσα σειρά, χρησιμοποιώντας τη μέθοδο της ταξινόμησης φυσαλίδας (bubble sort). Θα πρέπει να έχει χρονική πολυπλοκότητα O(N) στην καλύτερη περίπτωση και $O(N^2)$ στη χειρότερη περίπτωση, όπου N το πλήθος των στοιχείων της λίστας, και χωρική πολυπλοκότητα O(1) — δηλαδή η ταξινόμηση θα πρέπει να γίνεται επί τόπου, δεν επιτρέπεται π.χ. να αντιγράψετε τα στοιχεία της λίστας σε ένα πίνακα.

5. (12)

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

Θεωρούμε έναν μονοδιάστατο πίνακα **a** που περιέχει **N** ακέραιους αριθμούς, τα στοιχεία του οποίου είναι ταξινομημένα σε αύξουσα σειρά: $a_1 \le a_2 \le \ldots \le a_N$. Το *κόστος του στοιχείου που βρίσκεται στη θέση i του* πίνακα, όπου $1 \le i \le N$, δίνεται από τον τύπο:

$$cost(i) = \sum_{j=1}^{N} |a_j - a_i|$$

Να γράψετε μία <u>*καλιμή και αποδοτική*</u> συνάρτηση που δέχεται ως παραμέτρους τα **N**, **a** και έναν μονοδιάστατο πίνακα **c** με χώρο για **N** στοιχεία. Η συνάρτησή σας πρέπει να υπολογίζει τις τιμές του κόστους των στοιχείων και να τις αποθηκεύει στις αντίστοιχες θέσεις του πίνακα **c**.

<u>Παράδειγμα 1</u>: (**N** = 4)
a = [0, 0, 1, 2]
c = [3, 3, 3, 5]

<u>Παράδειγμα 2</u>: (**N** = 7)
a = [2, 3, 5, 7, 11, 13, 17]
c = [44, 39, 33, 31, 35, 41, 61]