ΤΕΜ-101 Εισαγωγή στους Η/Υ – Εξεταστική Ιανουαρίου 2011

Θέματα Β

1. **(10 μον.)** Απαντήστε σωστό ή λάθος στις παρακάτω ερωτήσεις

   (α′) `_2togo` είναι έγκυρο όνομα μεταβλητής

   (β′) Αν `p` είναι δείκτης στο πρώτο στοιχείο ενός πίνακα τότε `p[1]` είναι δείκτης στο δεύτερο στοιχείο του πίνακα

   (γ′) Αν `p` είναι δείκτης στο πρώτο στοιχείο ενός πίνακα τότε `*p+1` είναι η τιμή του δεύτερου στοιχείου του πίνακα

   (δ′) Αν `int x = 5, y = 4, z = 3;` τότε μετά την εκτέλεση της εντολής `y += ++z-y/x--;` και οι τρείς μεταβλητές έχουν την ίδια τιμή

   (ε′) Η ανακύκλωση `for (i = 21; i > 0 && (i % 3 == 0 || i  % 7 == 0); i--) {...}` εκτελείται ακριβώς 2 φορές

   Απάντηση.

   (α′) Σωστό. Ο πρώτος χαρακτήρας του ονόματος μιας μεταβλητής μπορεί να είναι το χαρακτήρας '_'.

   (β′) Λάθος. Μια και `p[1]` είναι ισοδύναμο με `*(p+1)`, η έκφραση που δίδεται είναι η τιμή του δεύτερου στοιχείου του πίνακα στον οποίο δείχνει ο δείκτης `p`.

   (γ′) Λάθος. `*p+1` είναι η τιμή του πρώτου στοιχείου του πίνακα στον οποίο δείχνει ο `p` αυξημένη κατά ένα.

   (δ′) Λάθος. Από την προτεραιότητα των τελεστών που εμφανίζονται προκύπτει ότι η εντολή ανάθεσης που δίδεται είναι ισοδύναμη με τις `++z; y = y + z - y/x; x--;`. Επομένως μετά την εκτέλεση των εντολών έχουμε `x = 4`, `y = 8` και `z = 4`.

   (ε′) Λάθος. Η ανακύκλωση θα εκτελεστεί όταν `i = 21` αλλά όχι όταν η μεταβλητή `i` πάρει την τιμή 20.

2. **(10 μον.)** Γράψτε τη συνάρτηση `int numdiv(int m,  int n)` η οποία υπολογίζει και επιστρέφει το πλήθος των αριθμών στο διάστημα $[m, n]$ οι οποίοι είναι πολλαπλάσια του 3 ή του 5. Υποθέτουμε εδώ ότι $1 \leq m < n$.

   Απάντηση.

```
int numdiv(int m, int n)
{
    int i, s = 0;

    for (i = m; i <= n; i++)
        if (i % 3 == 0 || i % 5 == 0) s++;

    return s;
}
```

3. **(10 μον.)** Τι τυπώνει η εντολή `printf` στο παρακάτω πρόγραμμα;

```
int x = 1, y, z;

while (x < 10) {
    y = x++;
    z = ++x;
}
printf("x = %d y = %d z = %d\n", x, y, z);
```

Απάντηση. Από τις ιδιότητες του τελεστή μοναδιαίας αύξησης συμπεραίνουμε ότι οι εντολές στο σώμα της ανακύκλωσης είναι ισοδύναμες με τις `y = x; x += 2; z = x;`. Επομένως οι διαδοχικές τιμές των μεταβλητών `x, y, z` είναι:

```
3, 1, 3
5, 3, 5
7, 5, 7
9, 7, 9
11, 9, 11
```

και η εντολή `printf` θα τυπώσει: `x = 11 y = 9 z = 11`.

4. **(5 μον.)** Πόσα στοιχεία έχει ο πίνακας `int a[201]`; Ποιό είναι το πρώτο στοιχείο του; Ποιό είναι το τελευταίο; Υπάρχει κάποιο πρόβλημα στην παρακάτω ανακύκλωση;

```
for (i = 1; i <= 201; i++) a[i] = 0;
```

Γράψτε μια ανακύκλωση η οποία αναθέτει στα στοιχεία του πίνακα τις τιμές $0, 1, 2, 0, 1, 2, \ldots$.

Απάντηση. Ο πίνακας `a` έχει, προφανώς, 201 στοιχεία. Το πρώτο στοιχείο είναι το `a[0]` και το τελευταίο το `a[200]`, μια και η αρίθμηση στη C ξεκινάει από το μηδέν. Το πρόβλημα στην ανακύκλωση που δίδεται στην εκφώνηση της άσκησης είναι ότι στην τελευταία επανάληψη της ανακύκλωσης εκτελείται η εντολή `a[201] = 0`. Το τελευταίο όμως στοιχείο του πίνακα είναι το `a[200]`. Ένα άλλο, ίσως, πρόβλημα είναι ότι δεν αρχικοποιείται το στοιχείο `a[0]`. Τέλος, η ανακύκλωση

```
for (i = 0; i < 201; i += 3)  {
    a[i] = 0;
    a[i+1] = 0;
    a[i+2] = 0;
}
```

αναθέτει στα στοιχεία του πίνακα `a` τις τιμές $0, 1, 2, 0, 1, 2, \ldots$.

5. **(10 μον.)** Τι τυπώνει το παρακάτω πρόγραμμα;

```
int main()
{
    int i, a[] = {3, 5, 6, 7, 7, 7};
    int *p = a, *q = a;

    for (i = 0; i < 5; i++) {
        printf("%d ", *q);  ++p;
    }
```

```
    for (i = 0; i < 5; i++) {
        printf("%d ", *q);  ++q;
    }

    return 0;
}
```

Απάντηση. Παρατηρήστε ότι στην πρώτη ανακύκλωση, δεν αλλάζει τη τιμή του δείκτη `q`. Το πρόγραμμα λοιπόν θα τυπώσει: `3 3 3 3 3 3 5 6 7 7`.

6. **(15 μον.)** Γράψτε ένα πρόγραμμα το οποίο διαβάζει ένα ακέραιο αριθμό δευτερολέπτων και τυπώνει τον ισοδύναμο χρόνο σε ώρες, λεπτά και δευτερόλεπτα. Για παράδειγμα, αν η είσοδος είναι ο αριθμός 7322 το πρόγραμμά σας πρέπει να τυπώσει:

```
7322 seconds is 2 hours 2 minutes 2 seconds
```

Απάντηση.

```
#include <stdio.h>

int main()
{
    int h, m, s;

    printf("Enter number of seconds: ");
    scanf("%d", &s);
    printf("%d seconds is ", s);

    h = s / 3600;  /* Number of hours in s */
    s -= h*3600;  /* What is left is minutes and seconds */
    m = s / 60;    /* Number of minutes in s */
    s -= m*60;    /* What is left is just seconds */

    printf("%d hours %d minutes %d seconds\n", h, m, s);
    return 0;
}
```

7. **(20 μον.)** Γράψτε τη συνάρτηση `void big2(int a[], int n)` η οποία βρίσκει και τυπώνει τα **δύο** μεγαλύτερα στοιχεία του πίνακα ακεραίων `a` μήκους `n`.

Απάντηση.

```
void big2(int a[], int n)
{
    int i, pri, sec;

    pri = sec = a[0];

    for (i = 1; i < n; i++) {
        if (a[i] > pri) {
            sec = pri;
            pri = a[i];
```

```
        }
        else if (a[i] > sec && a[i] < pri) {
            sec = a[i];
        }
    }
    printf("Largest and second largest elements: %d %d\n", pri, sec);
}
```

8. **(10 μον.)** Μετατρέψτε τους αριθμούς 98, 765, 4321 στο οκταδικό και δεκαεξαδικό σύστημα.

   Απάντηση. Έχουμε ότι $98 = 1 \times 8^2 + 4 \times 8^1 + 2$, επομένως $(98)_{10} = (142)_8$. Επίσης, $98 = 6 \times 16^1 + 2$, άρα $(98)_{10} = (62)_{16}$. Τώρα, $765 = 1 \times 8^3 + 3 \times 8^2 + 7 \times 8^1 + 5$ άρα $(765)_{10} = (1375)_8$. Επιπλέον, $765 = 2 \times 16^2 + 15 \times 16^1 + 13$, άρα $(765)_{10} = (2fd)_{16}$. Τέλος, $4321 = 1 \times 8^4 + 3 \times 8^2 + 4 \times 8^1 + 1$, άρα $(4321)_{10} = (10341)_8$. Ακόμα, $4321 = 1 \times 16^3 + 14 \times 16^1 + 1 \times 16^0$ άρα $(4321)_{10} = (10e1)_{16}$.

9. **(10 μον.)** Τι τυπώνει το παρακάτω πρόγραμμα;

```
int A(int i)
{
    printf("A\n");
    return 2*i+1;
}

double B(char c)
{
    printf("B");
    return 7;
}

void C(double x)
{
    printf("C\n");
}

int main()
{
    int p = 0;
    double q = 0.0;
    char r = 'W';

    p = A(p);
    q = B(r);
    C(B(r));
    p = A(A(p));

    return 0;
}
```

   Απάντηση. Το πρόγραμμα θα τυπώσει:

```
A
BBC
A
A
```

Κάθε κλήση της συνάρτησης `A` τυπώνει τον χαρακτήρα `A` και αλλάζει γραμμή, κάθε κλήση της συνάρτησης `B` τυπώνει ένα χαρακτήρα `B` και κάθε κλήση της συνάρτησης `C` τυπώνει ένα χαρακτήρα `C` και αλλάζει γραμμή.