ΤΕΜ-102 Προγραμματισμός Η/Υ – Εξεταστική Ιουνίου 2011

Διάρκεια εξέτασης: 3 ώρες. Καλή επιτυχία!

A

1	2	3	4	5	6	7	8	9

1. **(5 μον.)** Τι τυπώνει το παρακάτω πρόγραμμα;

```
void f(int x)
{
    printf("x = %d\n", x);
    x = 103;
    printf("x = %d\n", x);
}

int main()
{
    int x = 102;
    printf("x = %d\n", x);
    f(x);
    printf("x = %d\n", x);
    return 0;
}
```

Απάντηση: Στην έξοδο του προγράμματος θα δούμε διαδοχικά

```
x = 102   /* First printf of main() */
x = 102   /* First printf of f() */
x = 103   /* Second printf of f() */
x = 102   /* Second printf of main() */
```

Θυμηθείτε ότι τα ορίσματα μιας συνάρτησης **αντιγράφονται** στα λεγόμενα τυπικά ορίσματα της συνάρτησης κατά την κλίση της.

2. **(15 μον.)** Γράψτε τη συνάρτηση `void rshift(int *a, int n, int shift)` η οποία μετακινεί `shift` θέσεις δεξιά τα στοιχεία του πίνακα `a` μήκους `n`. Οι κενές θέσεις συμπληρώνονται με μηδενικά. Για παράδειγμα, αν ο πίνακας `a` περιέχει τα στοιχεία 3, 89, 21, 7, 33, 12 μετά τη κλήση `rshift(a, 6, 3)` τα στοιχεία του πίνακα `a` θα πρέπει να είναι 0, 0, 0, 3, 89, 21. Υποθέστε ότι `shift` είναι μεγαλύτερο ή ίσο του μηδενός.

Απάντηση:

```
void rshift(int *a, int n, int shift)
{
    int i;

    for (i = n-1; i >= shift; i--) a[i] = a[i-shift];
    if (shift > n) shift = n;
    for (i = 0; i < shift; i++) a[i] = 0;

}
```

Η πρώτη ανακύκλωση μετακινεί τα στοιχεία `a[0]`, `a[1]`, ..., `a[shift-1]` στη σωστή τους θέση ενώ η δεύτερη ανακύκλωση τα μηδενίζει.

3. **(10 μον.)** Περιγράψτε τι ακριβώς δηλώνεται στις παρακάτω δηλώσεις

   (α′) `char *s;`

   (β′) `int *p[3];`

   (γ′) `char (*q)[4];`

   (δ′) `int (*f)(int, int);`

   Απάντηση: Στην πρώτη δήλωση, `s` είναι δείκτης σε `char`. Στη δεύτερη δήλωση `p` είναι πίνακας μήκους 3 τα στοιχεία του οποίου είναι δείκτες σε `int`. Στην τρίτη δήλωση `q` είναι δείκτης σε πίνακα μήκους 4 τα στοιχεία του οποίου είναι `char`. Στην τελευταία δήλωση, `f` είναι δείκτης σε συνάρτηση με δύο ορίσματα τύπου `int`.

4. **(10 μον.)** Υποθέτουμε ότι ένας πίνακας χαρακτήρων `s` περιέχει ακριβώς δύο λέξεις χωρισμένες από ένα μοναδικό κενό χαρακτήρα. Γράψτε κατάλληλες εντολές οι οποίες αντιστρέφουν τη σειρά των λέξεων στον πίνακα χαρακτήρων. Για παράδειγμα, ”Maria Nikos” θα πρέπει να γίνει ”Nikos Maria”. Μπορείτε να υποθέσετε οτι οι δύο λέξεις έχουν **ίσο** αριθμό χαρακτήρων.

   Απάντηση

```
char c, *p, *q;

for (p = s; *p != ' '; p++); /* Find the space */
p++; /* Move to the first charcater of second word */
for (q = s; *q; q++) {  /* Swap characters */
    c = *q;  *q = *p; *p = c;
    p++;
}
```

5. **(20 μον.)** Ορίστε μια κατάλληλη δομή `struct rect` για την περιγραφή ενός παραλληλόγραμμου. Γράψτε μια συνάρτηση η οποία με ορίσματα δύο `struct rect` επιστρέφει ένα αν τα δύο παραλληλόγραμμα τέμνονται, μηδέν διαφορετικά.

   Απάντηση Είναι μάλλον πιο εύκολο να δεί κανείς πότε δύο παραλληλόγραμμα **δεν** τέμνονται:

```
struct rect {
    double xl, yb; /* Coordinates of south-west corner */
    double xr, yt;  /* Coordinates of north-east corner */
};

int Intersect(struct rect r, struct rect s)
{
    return !( s.yb > r.yt || s.yt < r.yb || s.xr < r.xl || s.xl > r.xr );
}
```

6. **(5 μον.)** Αν `p` είναι δείκτης στο πρώτο στοιχείο ενός πίνακα `a` με στοιχεία 1, 0, 2, 1, 0, 2, τι τυπώνουν οι παρακάτω εντολές;

```
printf("%d %d %d\n", a[0], *p, ++*p);
p = a + 5;
for (i = 1; i < 4; i++) printf("%d %d\n", a[i], *(p-i));
```

Απάντηση: Το πρόγραμμα θα τυπώσει, κατά σειρά:

```
2 2 2
0 0
2 1
1 2
```

Παρατηρήστε ότι αφού αρχικά `p = a`, η εντολή `++*p` θα αυξήσει κατά ένα την τιμή του στοιχείου στο οποίο δείχνει ο δείκτης `p`, δηλαδή του `a[0]`, το οποίο είχε την τιμή ένα.

7. **(10 μον.)** Γράψτε τη συνάρτηση `int IsSorted(double *a, int n)` η οποία επιστρέφει ένα αν το διάνυσμα `a` μήκους `n` είναι ταξινομημένο κατά αύξουσα σειρά, μηδέν διαφορετικά.

   Απάντηση

```
int IsSorted(double *a, int n)
{
    int i;

    for (i = 1; i < n; i++)
        if (a[i] < a[i-1]) return 0;

    return 1;
}
```

8. **(15 μον.)** Τι τυπώνει το παρακάτω πρόγραμμα;

```
struct tem {
    char *s;
    int i;
    struct tem *next;
};

int main()
{
    struct tem a[] = {{"Mary", 1, a+1}, {"Jane", 2, a+2}, {"Nick", 3, a}};
    struct tem *p = a;
    int i;

    printf("%s %s %s\n", a[0].s, p->s, a[2].next->s);

    for (i = 0; i < 2; i++) printf("%d %c\n", --a[i].i, ++a[i].s[3]);

    printf("%s %s %s\n", p->s, a[p->i].s, a[--(p->next->i)].s);

    return 0;
}
```

Απάντηση Το πρόγραμμα θα εκτυπώσει κατ' αρχήν

```
Mary Mary Mary
```

και μετά ένα μήνυμα σφάλματος διότι η έκφραση `++a[i].s[3]` προσπαθεί να αλλάξει την τιμή μιας σταθεράς τύπου χαρακτήρα. Αν δέν υπήρχε το συγκεκριμένο πρόβλημα, η τελευταία εντολή `printf` του προγράμματος θα τύπωνε

```
Mary Jane Jane
```

9. **(10 μον.)** Γράψτε μια συνάρτηση η οποία με ορίσματα ένα πίνακα `a` με `m` γραμμές και `n` στήλες υπολογίζει και επιστρέφει την ποσότητα

$$\max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$$

Υποθέστε ότι ο πίνακας έχει μέγιστη διάσταση στηλών `MAXCOLS` και ότι τα στοιχεία του είναι τύπου `double`. Χρησιμοποιήστε τη συνάρτηση `fabs()` για τον υπολογισμό της απόλυτης τιμής.

Απάντηση

```
double myfunc(double a[][MAXCOLS], int m, int n)
{
    double s, r = 0.0;
    int i, j;

    for (i = 0; i < m; i++) {
        s = 0.0;
        for (j = 0; j < n; j++) s += fabs(a[i][j]);
        if (s > r) r = s;
    }

    return r;
}
```