ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2008 (5/2/2008)

1. (α΄) Ποιος είναι ο στόχος των διαδικασιών της μεταγλώττισης (compilation) και της σύνδεσης (linking) στον προγραμματισμό;

   (β΄) Ποια πλεονεκτήματα προσφέρει η χρήση συναρτήσεων στον προγραμματισμό; Θα μπορούσε να υπάρχει και κάποιο μειονέκτημα στη χρήση πολλών συναρτήσεων και, αν ναι, συγκρίνετέ το με τα πλεονεκτήματα που αναφέρατε προηγουμένως. Δηλαδή, κατά τη γνώμη σας, υπερισχύει ή όχι;

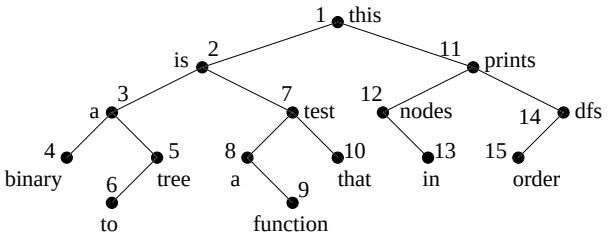
   (γ΄) Σε σχέση με τη γλώσσα προγραμματισμού C, με ποιους τρόπους μπορεί μία συνάρτηση να επιστρέψει το αποτέλεσμα (ή τα αποτελέσματα) που υπολόγισε στη συνάρτηση που την κάλεσε; Θα μπορούσαν να χρησιμοποιηθούν εξωτερικές/καθολικές μεταβλητές για τον σκοπό αυτό και, αν ναι, πώς κρίνετε αυτή την προσέγγιση;

   (δ΄) Είναι συμφέρον ή όχι, και γιατί, να διασπάσουμε ένα πολύ μεγάλο πρόγραμμα που είναι δομημένο σε συναρτήσεις σε πολλά αρχεία; Σε σχέση με τη γλώσσα προγραμματισμού C, σε τι χρησιμεύουν τα αρχεία επικεφαλίδας (header files) και συνήθως τι καταχωρούμε σε αυτά;

   Απαντήστε στα προηγούμενα συνοπτικά, *σε 20 το πολύ γραμμές*.

2. Στην C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου συμβολοσειρών ως εξής:

```
typedef struct treenode *Treeptr;

struct treenode {
  char *str;
  Treeptr left;
  Treeptr right;
};
```



   (α΄) Γράψτε σε C μία συνάρτηση που να παίρνει σαν όρισμα ένα δυαδικό δέντρο συμβολοσειρών και να εκτυπώνει τους κόμβους του κατά σειρά, σύμφωνα με μία *πρώτα-κατά-βάθος* διάσχισή του. Στο παραπάνω σχήμα δεξιά, οι αύξοντες αριθμοί στους κόμβους του δέντρου υποδεικνύουν πώς ορίζεται αυτή η σειρά διάσχισης. Για παράδειγμα, αν δίναμε στη συνάρτηση που θα γράψετε το δέντρο του σχήματος, αυτή θα έπρεπε να εκτυπώσει:

```
this is a binary tree to test a function that prints nodes in dfs order
```

   (β΄) Πώς θα ορίζατε το πρωτότυπο μίας συνάρτησης η οποία θα ήταν σε θέση, δεδομένης μίας συμβολοσειράς και ενός **ταξινομημένου** δυαδικού δέντρου συμβολοσειρών, να διαγράψει από το δέντρο τον κόμβο που το περιεχόμενό του ταυτίζεται με τη συμβολοσειρά αυτή, εφόσον υπάρχει τέτοιος κόμβος; Η συνάρτηση αυτή θα πρέπει να πληροφορεί με κάποιο τρόπο την καλούσα συνάρτηση τόσο για το αν βρέθηκε σχετικός κόμβος, όσο και για το ποιο είναι το νέο δέντρο μετά την πιθανή διαγραφή του κόμβου. Χωρίς να δώσετε την υλοποίηση σε C της συνάρτησης, περιγράψτε, σε γενικές γραμμές, πώς θα αντιμετωπίζατε το πρόβλημα, λαμβάνοντας φυσικά υπόψη ότι στην περίπτωση που διαγραφεί κόμβος, θα πρέπει το νέο δέντρο να είναι επίσης ταξινομημένο.

3. Αποδεικνύεται μαθηματικά ότι

$$\frac{\pi}{2} = \frac{2}{1} \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \frac{8}{7} \times \frac{8}{9} \ldots$$

   όπου π είναι το γνωστό μας $3.14\ldots$. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`pibyprod.c`"), το οποίο να παίρνει από τη γραμμή εντολής ένα θετικό ακέραιο N και να υπολογίζει προσεγγιστικά την τιμή του π, με βάση τον προηγούμενο τύπο, πολλαπλασιάζοντας N όρους του γινομένου. Κάποια παραδείγματα εκτέλεσης δίνονται στην επόμενη σελίδα.

```
% ./pibyprod 1000000
Multiplied 1000000 terms, pi is 3.1415910828
% ./pibyprod 100000000
Multiplied 100000000 terms, pi is 3.1415926379
```

Δεδομένου ότι η τιμή του π με ακρίβεια 10 δεκαδικών ψηφίων είναι 3.1415926536, οπότε τα προηγούμενα αποτελέσματα απέχουν απ΄ αυτή, πιστεύετε ότι θα αρκούσε, για να επιτύχουμε οσοδήποτε μεγάλη ακρίβεια θα θέλαμε, να εκτελέσουμε το πρόγραμμα με αρκετά μεγάλο N; Γνωρίζετε ότι οι πραγματικοί αριθμοί είναι άπειροι. Είναι δυνατόν οποιοσδήποτε απ΄ αυτούς να καταχωρηθεί σ΄ ένα υπολογιστή; Αν όχι, πόσοι περίπου είναι αυτοί που μπορούν να καταχωρηθούν, και από τι εξαρτάται το πλήθος τους;

4. Στο "*Σύλλογο Φίλων του Προγραμματισμού*" πρόκειται να γίνουν εκλογές για την ανάδειξη προέδρου, για την οποία θέση μπορεί να βάλει υποψηφιότητα οποιοδήποτε μέλος του συλλόγου. Το εκλογικό σύστημα που ισχύει, για την εκλογή προέδρου, με βάση το καταστατικό του συλλόγου είναι λίγο ιδιόρρυθμο. Συγκεκριμένα, κάθε μέλος του συλλόγου δεν ψηφίζει απλώς τον υποψήφιο της προτίμησής του, αλλά διατάσσει όλους τους υποψήφιους σε φθίνουσα σειρά προτίμησης. Ο τρόπος με τον οποίο εκλέγεται ο πρόεδρος είναι ο εξής. Αρχικά, λαμβάνονται υπόψη μόνο οι πρώτες προτιμήσεις των ψηφοφόρων. Αν με βάση αυτές, κάποιος υποψήφιος συγκεντρώνει ποσοστό **μεγαλύτερο** του 50%, εκλέγεται αυτός. Αν όχι, ο υποψήφιος που έλαβε το μικρότερο ποσοστό ψήφων, διαγράφεται από υποψήφιος και οι ψήφοι που έλαβε πηγαίνουν στις αμέσως επόμενες προτιμήσεις των ψηφοφόρων που τον είχαν ψηφίσει. Η διαδικασία διαγραφής του υποψηφίου που μειοψηφεί και η μεταφορά των ψήφων που έλαβε στις επόμενες επιλογές των ψηφοφόρων του επαναλαμβάνεται μέχρι κάποιος υποψήφιος να πάρει την απόλυτη πλειοψηφία, δηλαδή ποσοστό μεγαλύτερο του 50%. Ένας επιπλέον κανόνας που ισχύει επίσης είναι ότι όταν μειοψηφούν εξ ίσου περισσότεροι του ενός υποψήφιοι, διαγράφεται εκείνος που έγινε μέλος στο σύλλογο πιο πρόσφατα. Δηλαδή, στην περίπτωση αυτή υπερισχύουν οι αρχαιότεροι υποψήφιοι.

   Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`elections.c`"), το οποίο να καλείται με δύο ορίσματα στη γραμμή εντολής, το πλήθος των ψηφοφόρων NV και το πλήθος των υποψηφίων NC και να διαβάζει από την πρότυπη είσοδο (`stdin`) τις προτιμήσεις των ψηφοφόρων. Σε κάθε γραμμή της εισόδου δίνονται οι επιλογές ενός ψηφοφόρου για τους υποψηφίους, σε φθίνουσα σειρά προτίμησης. Κάθε υποψήφιος παριστάνεται από έναν αριθμό, από 1 έως NC, ο οποίος δείχνει και την αρχαιότητα του υποψηφίου (ο υποψήφιος 1 είναι ο αρχαιότερος όλων). Υποθέστε ότι τα δεδομένα που δίνονται στο πρόγραμμα είναι σωστά, δηλαδή κάθε γραμμή της εισόδου είναι μία μετάθεση των αριθμών από το 1 έως το NC. Το πρόγραμμά σας να βρίσκει και να εκτυπώνει τον τελικό νικητή, αλλά και τα ενδιάμεσα αποτελέσματα, για κάθε γύρο της διαδικασίας. Ενδεικτικές εκτελέσεις είναι οι εξής:

```
% cat votes.txt
3 2 4 1
2 1 4 3
2 1 3 4
4 3 1 2
2 1 3 4
1 2 3 4
1 3 4 2
3 2 4 1
2 1 3 4
4 3 1 2
3 2 4 1
2 1 3 4
4 1 2 3
%
```

```
% ./elections 13 4 < votes.txt
Round 1 results:  15.38%  38.46%  23.08%  23.08%
Candidate 1 eliminated
Round 2 results:   0.00%  46.15%  30.77%  23.08%
Candidate 4 eliminated
Round 3 results:   0.00%  53.85%  46.15%   0.00%
Final winner is candidate 2
%
% ./elections 12 4 < votes.txt
Round 1 results:  16.67%  41.67%  25.00%  16.67%
Candidate 4 eliminated
Round 2 results:  16.67%  41.67%  41.67%   0.00%
Candidate 1 eliminated
Round 3 results:   0.00%  50.00%  50.00%   0.00%
Candidate 3 eliminated
Round 4 results:   0.00% 100.00%   0.00%   0.00%
Final winner is candidate 2
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2009 (14/2/2009)

1. (α΄) Στις εξετάσεις κάποιου μαθήματος προγραμματισμού ζητήθηκε από τους εξεταζόμενους φοιτητές να γράψουν μία συνάρτηση C, η οποία θα διαβάζει από την πρότυπη είσοδο (stdin) την απάντηση του χρήστη σε μία ερώτηση και αν αυτή είναι `yes`, να επιστρέφει με κάποιο τρόπο την τιμή 1, ενώ αν είναι οτιδήποτε άλλο, να επιστρέφει την τιμή 0. Κάποιοι φοιτητές έδωσαν τις απαντήσεις που φαίνονται παρακάτω.

```
void get_reply(int rep)
{  char *str;
   scanf("%s ", str);
   if (str = "yes")
      rep = 1;
   else
      rep = 0;
}
```

```
int *get_reply(void)
{  char *str;
   int rep = 0;
   scanf("%s ", &str);
   if (str == "yes")
      rep = 1;
   return &rep;
}
```

Οι δύο παραπάνω συναρτήσεις, αν και είναι συντακτικά σωστές, έχουν σοβαρά λάθη. Ποια είναι αυτά, γιατί είναι λάθη και πώς θα γράφατε εσείς τη ζητούμενη συνάρτηση;

(β΄) Σε ένα μεγάλο πρόγραμμα C, εκτός από τη συνάρτηση `main`, υπάρχουν και

i. οι συναρτήσεις `strprocA`, `strprocB` και `strprocC`, που διαχειρίζονται συμβολοσειρές και χρησιμοποιούν κάποια συμβολική σταθερά `STRCONST`, καθώς και

ii. οι συναρτήσεις `arrprocA` και `arrprocB`, που διαχειρίζονται πίνακες και χρησιμοποιούν κάποιες συμβολικές σταθερές `ARRCONST1` και `ARRCONST2`.

Η συνάρτηση `main` καλεί όλες τις παραπάνω συναρτήσεις και χρειάζεται και τις συμβολικές σταθερές που χρησιμοποιούν και αυτές. Οι συναρτήσεις για τις συμβολοσειρές μπορεί να καλούν η μία την άλλη και το ίδιο ισχύει και για τις συναρτήσεις για τους πίνακες. Όμως, δεν υπάρχει ανάγκη επικοινωνίας μεταξύ των συναρτήσεων της πρώτης και της δεύτερης κατηγορίας. Πώς θα διασπούσατε το πρόγραμμα αυτό σε αρχεία (πηγαία και επικεφαλίδας) με τον πλέον ιδανικό τρόπο και τι θα περιλαμβάνατε στο καθένα απ΄ αυτά τα αρχεία;

2. Για να χρησιμοποιήσουμε στην C μία λίστα ακεραίων, συνήθως ορίζουμε τον κόμβο της ως εξής:

```
typedef struct listnode *Listptr;
struct listnode {
   int value;
   Listptr next; }
```

(α΄) Γράψτε δύο εκδοχές (μία αναδρομική και μία επαναληπτική) μίας συνάρτησης C, η οποία θα παίρνει σαν παράμετρο μία λίστα ακεραίων και θα απελευθερώνει (μέσω της συνάρτησης `free`) τη μνήμη που είχε δεσμευθεί για τους κόμβους της λίστας.

(β΄) Γράψτε μία συνάρτηση C που να διαβάζει από την πρότυπη είσοδο (stdin) ακεραίους, μέχρι το τέλος της εισόδου, και να επιστρέφει με κάποιο τρόπο μία λίστα με αυτούς τους ακεραίους, με αντίστροφη σειρά από αυτήν που εισήχθησαν. Δεν επιτρέπεται να χρησιμοποιήσετε κάποια από τις γνωστές σας συναρτήσεις διαχείρισης λιστών. Επίσης, υποθέστε ότι οποιαδήποτε απόπειρα δυναμικής δέσμευσης μνήμης από τη συνάρτησή σας θα είναι επιτυχημένη.

(γ΄) Δώστε **μόνο** το πρωτότυπο μίας συνάρτησης C που θα προσαρτά μία λίστα στο τέλος μίας άλλης.

(δ΄) Δώστε **μόνο** το πρωτότυπο μίας συνάρτησης C που θα εισάγει στη θέση N μίας λίστας ακεραίων έναν ακέραιο x.

3. Η εικασία[1] του Goldbach λέει ότι κάθε άρτιος ακέραιος αριθμός μεγαλύτερος του 2 μπορεί να γραφεί σαν άθροισμα δύο πρώτων αριθμών. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`goldbach.c`"), το οποίο να καλείται με έναν ακέραιο N σαν όρισμα στη γραμμή εντολής και να εκτυπώνει όλους τους τρόπους με τους οποίους κάθε άρτιος θετικός ακέραιος μικρότερος ή ίσος του N μπορεί να γραφεί σαν άθροισμα δύο πρώτων αριθμών. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
% ./goldbach 100
4 = 2 + 2
6 = 3 + 3
8 = 3 + 5
10 = 3 + 7
10 = 5 + 5
12 = 5 + 7
14 = 3 + 11
14 = 7 + 7
...................
98 = 19 + 79
98 = 31 + 67
98 = 37 + 61
100 = 3 + 97
100 = 11 + 89
100 = 17 + 83
100 = 29 + 71
100 = 41 + 59
100 = 47 + 53
%
```

4. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`maxsum.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (stdin) τις διαστάσεις ενός διδιάστατου πίνακα (πλήθος γραμμών και στηλών) και να δημιουργεί έναν τέτοιον πίνακα με στοιχεία τυχαίους ακεραίους αριθμούς[2] στο διάστημα `[-MAXNUM,MAXNUM]` (ορίστε το `MAXNUM` σαν μία συμβολική σταθερά με τιμή, π.χ., 99). Στη συνέχεια, το πρόγραμμά σας να βρίσκει και να εκτυπώνει εκείνον τον υποπίνακα του αρχικού πίνακα που έχει το μεγαλύτερο άθροισμα στοιχείων από όλους τους δυνατούς υποπίνακες του αρχικού. Αν υπάρχουν περισσότεροι του ενός υποπίνακες με το ίδιο μέγιστο άθροισμα, να εκτυπώνεται οποιοσδήποτε από αυτούς. Επισημαίνεται ότι ακόμα και ένα στοιχείο του αρχικού πίνακα αποτελεί υποπίνακά του, όπως και ολόκληρος ο αρχικός πίνακας είναι υποπίνακας του εαυτού του. Κάποιο παράδειγμα εκτέλεσης:

```
% ./maxsum
Please give the dimensions of the matrix: 6 10
   49   49   51   91    3  -37  -87  -43  -56   16
  -83  -88   63   29   90   98   39  -48   80  -88
   10  -44  -54   -2  -50   10    4   76   48  -73
   32   15  -98   19  -32   87   57   64   -1    7
   32   44   88  -86  -31  -94   18  -98  -27  -48
   92   24  -42   10   94  -57  -82   77  -93   85
Maximum sum is 568 in submatrix from (1,3) to (3,8):
   29   90   98   39  -48   80
   -2  -50   10    4   76   48
   19  -32   87   57   64   -1
%
```

[1] Εικασία είναι μία μαθηματική πρόταση που δεν έχει αποδειχθεί, αλλά πιστεύεται ότι είναι σωστή.

[2] Υπενθυμίζεται ότι η συνάρτηση `rand()` της C επιστρέφει έναν τυχαίο ακέραιο αριθμό από το 0 μέχρι κάποιο μέγιστο, που μπορεί να είναι διαφορετικό από σύστημα σε σύστημα.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2012 (13/2/2012)

1. (α΄) Υποθέστε ότι συμμετέχετε, ως συνεργάτες του μαθήματος, στο φόρουμ υποστήριξης (`lists`) που υπάρχει διαθέσιμο. Πώς θα απαντούσατε στις παρακάτω ερωτήσεις συναδέλφων σας;
   i. *Ο compiler μου βγάζει segmentation fault. Τι φταίει;*
   ii. *Τρέχω το πρόγραμμά μου και παίρνω segmentation fault. Καμία ιδέα;*
   iii. *Το πρόγραμμά μου τρέχει κανονικά στο Dev-C++, αλλά στο Linux μου βγάζει segmentation fault. Πού θα διορθωθεί η άσκηση;*
   iv. *Πώς θα μεταφέρω το πρόγραμμά μου στο PuTTY;*
   v. *Πρέπει να βάλουμε σχόλια στα προγράμματά μας;*

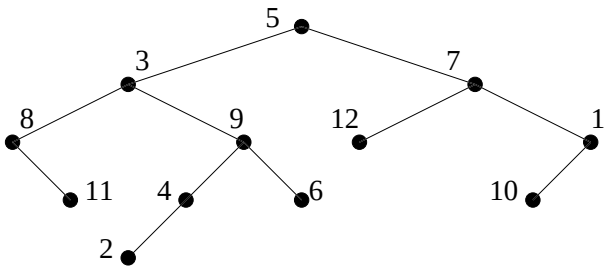
   (β΄) Χωρίς να αλλάξετε το πρωτότυπο της διπλανής συνάρτησης C, ξαναγράψτε τον κώδικα στο σώμα της, χρησιμοποιώντας το πολύ 15 χαρακτήρες, έτσι ώστε να διατηρήσει την ίδια ακριβώς λειτουργικότητα. Δικαιολογήστε την απάντησή σας.

```
unsigned int magic(unsigned int a, unsigned int b)
{  unsigned int s, c;
   for (s = c = 0 ; a ; a >>= 1, c++)
      if (a & 1)
         s += b << c;
   return s;
}
```

   (γ΄) Ορίστε μία μακροεντολή `ROUND(X)` για τον προεπεξεργαστή της C, μέσω `#define`, η οποία να υλοποιεί την στρογγύλευση ενός πραγματικού αριθμού στον πλησιέστερο ακέραιο.

2. Στην C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου ακεραίων ως εξής:

```
typedef struct treenode *Treeptr;

struct treenode {
  int value;
  Treeptr left;
  Treeptr right;
};
```



   (α΄) i. Υλοποιήστε δύο συναρτήσεις C με πρωτότυπα τα `int leftmost(Treeptr)`, η πρώτη, και `int rightmost(Treeptr)`, η δεύτερη, οι οποίες να παίρνουν σαν όρισμα ένα **μη κενό** δυαδικό δέντρο ακεραίων και να επιστρέφουν στο όνομά τους το περιεχόμενο του πλέον αριστερού και του πλέον δεξιού κόμβου του δέντρου, αντίστοιχα. Για παράδειγμα, οι συναρτήσεις αυτές θα πρέπει να επιστρέφουν για το δέντρο του παραπάνω σχήματος το 8 και το 1, αντίστοιχα.

   ii. Αν το δυαδικό δέντρο ήταν ταξινομημένο, τι ιδιότητες έχουν ο πλέον αριστερός και ο πλέον δεξιός κόμβος του;

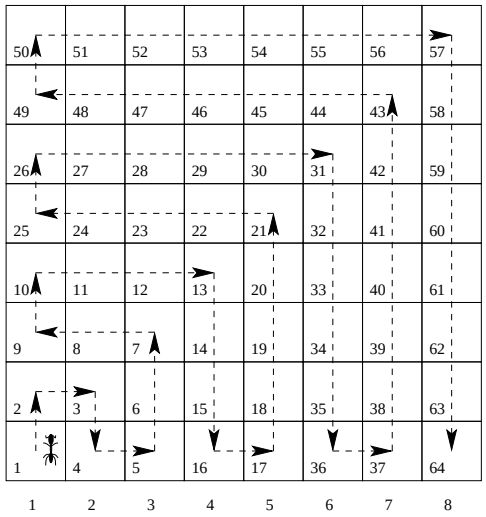
   iii. Αν θέλατε να υλοποιήσετε μία συνάρτηση C που να επιστρέφει ταυτόχρονα τα περιεχόμενα και του πλέον αριστερού και του πλέον δεξιού κόμβου ενός, **όχι κατ΄ ανάγκη μη κενού**, δυαδικού δέντρου ακεραίων, πώς θα ορίζατε το πρωτότυπό της;

   (β΄) Υλοποιήστε μία συνάρτηση C με πρωτότυπο `int pathssum(Treeptr, int)`, η οποία να παίρνει σαν όρισμα ένα δυαδικό δέντρο ακεραίων και έναν ακέραιο (έστω `sum`) και να επιστρέφει στο όνομά της το πλήθος των μονοπατιών που υπάρχουν στο δέντρο από τη ρίζα μέχρι κάποιο φύλλο, για τα οποία το άθροισμα των περιεχομένων των κόμβων τους ισούται με `sum`. Για παράδειγμα, αν η συνάρτηση αυτή καλείτο για το δέντρο του παραπάνω σχήματος και για `sum` ίσο με 23, θα έπρεπε να επιστρέψει το 3, γιατί υπάρχουν τρία ακριβώς μονοπάτια από τη ρίζα σε φύλλο που τα περιεχόμενα των κόμβων τους έχουν άθροισμα 23, τα 5-3-9-4-2, 5-3-9-6 και 5-7-1-10.

3. Δύο διαφορετικοί θετικοί ακέραιοι αριθμοί ονομάζονται *φίλιοι* (amicable) όταν το άθροισμα των διαιρετών του καθενός, εξαιρουμένου του εαυτού του, ισούται με τον άλλο. Για παράδειγμα, οι αριθμοί 220 και 284 είναι φίλιοι, αφού το άθροισμα των διαιρετών του 220, εξαιρουμένου του εαυτού του, δηλαδή των 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 και 110, ισούται με 284 και το άθροισμα των διαιρετών του 284, εξαιρουμένου του εαυτού του, δηλαδή των 1, 2, 4, 71 και 142, ισούται με 220. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`amicable.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (stdin) ένα θετικό ακέραιο N και να εκτυπώνει όλα τα ζευγάρια θετικών ακεραίων που αποτελούνται από φίλιους αριθμούς, μικρότερους ή ίσους του N, με τον πρώτο να είναι μικρότερος του δεύτερου. Μία ενδεικτική εκτέλεση του προγράμματος είναι η εξής:

```
% ./amicable
Please, give maximum number: 18000
220 - 284
1184 - 1210
2620 - 2924
5020 - 5564
6232 - 6368
10744 - 10856
12285 - 14595
%
```

4. Στο πρόβλημα του *μυρμηγκιού στη σκακιέρα* έχουμε ένα μυρμήγκι που ζει σε ένα ορθογώνιο πλαίσιο 8×8 (στη γενική περίπτωση $N \times N$). Το μυρμήγκι προτίθεται να επισκεφθεί όλον τον χώρο που ζει, αρχίζοντας από τη θέση με συντεταγμένες (1,1), προχωρώντας μετά περιφερειακά στα αμέσως γειτονικά τετραγωνίδια της αρχικής του θέσης, μετά, επίσης περιφερειακά, αλλά με αντίστροφη φορά, στα αμέσως γειτονικά αυτών των τετραγωνιδίων, κ.ο.κ. Κάθε χρονική στιγμή, το μυρμήγκι πηγαίνει και σε ένα νέο τετραγωνίδιο. Η ακριβής διαδρομή του μυρμηγκιού φαίνεται στο διπλανό σχήμα. Οι αριθμοί μέσα στα τετραγωνίδια είναι οι χρονικές στιγμές κατά τις οποίες το μυρμήγκι επισκέπτεται το καθένα από αυτά.



Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`ant.c`"), το οποίο να καλείται με έναν ακέραιο N σαν όρισμα στη γραμμή εντολής και να εκτυπώνει σε διάταξη πλαισίου $N \times N$ τις χρονικές στιγμές που επισκέπτεται το μυρμήγκι το κάθε τετραγωνίδιο, σύμφωνα με τη λογική που περιγράφηκε προηγουμένως. Ενδεικτικές εκτελέσεις του προγράμματος φαίνονται ακριβώς δεξιά.

```
% ./ant 8
  50   51   52   53   54   55   56   57
  49   48   47   46   45   44   43   58
  26   27   28   29   30   31   42   59
  25   24   23   22   21   32   41   60
  10   11   12   13   20   33   40   61
   9    8    7   14   19   34   39   62
   2    3    6   15   18   35   38   63
   1    4    5   16   17   36   37   64
% ./ant 3
   9    8    7
   2    3    6
   1    4    5
%
```

Bonus 10%: Αν δεν σας ήταν γνωστή η διάσταση N του πλαισίου μέσα στο οποίο ζει το μυρμήγκι, θα μπορούσατε να γράψετε ένα πρόγραμμα το οποίο να υπολογίζει τις συντεταγμένες του τετραγωνιδίου στο οποίο θα βρίσκεται το μυρμήγκι σε δοθείσα χρονική στιγμή; Για παράδειγμα, θα μπορούσε το πρόγραμμά σας να υπολογίζει ότι, ανεξαρτήτως του N, το μυρμήγκι θα πρέπει να βρίσκεται τη χρονική στιγμή 49271833 στο τετραγωνίδιο με συντεταγμένες (5472,7020); Δεν σας ζητείται να γράψετε το πρόγραμμα αυτό, αλλά να σκιαγραφήσετε, σε γενικές γραμμές, πώς θα το αντιμετωπίζατε. Σημειώστε ότι θα πρέπει το πρόγραμμα να κάνει τη μικρότερη δυνατή κατανάλωση μνήμης, αφού δεν χρειάζεται να υπολογίζει τις χρονικές στιγμές επίσκεψης του μυρμηγκιού σε κάθε τετραγωνίδιο, αλλά τις συντεταγμένες του τετραγωνιδίου που επισκέπτεται αυτό σε δεδομένη χρονική στιγμή.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2015 (4/2/2015)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση** αφαιρεί **1 μονάδα** από τον τελικό βαθμό.

1. Ο μεταγλωττιστής παράγει ...
   ◯ Α. ... πηγαία αρχεία
   ◯ Β. ... αντικειμενικά αρχεία
   ◯ Γ. ... εκτελέσιμα αρχεία

2. Οι γραμμές ενός προγράμματος C που αρχίζουν με # απευθύνονται προς ...
   ◯ Α. ... τον προεπεξεργαστή
   ◯ Β. ... το μεταγλωττιστή
   ◯ Γ. ... το συνδέτη

3. Ποια είναι η τιμή της αριθμητικής παράστασης `(39 / 4) % 5` στη C;
   ◯ Α. `1.95`
   ◯ Β. `0`
   ◯ Γ. `4`

4. Για την είσοδο δεδομένων σ΄ ένα πρόγραμμα C η γλώσσα διαθέτει κατάλληλες ...
   ◯ Α. ... εντολές
   ◯ Β. ... συναρτήσεις
   ◯ Γ. ... εντολές και συναρτήσεις

5. Η σταθερά χαρακτήρα `'0'` στη C ισούται με ...
   ◯ Α. ... τον ακέραιο αριθμό `0`
   ◯ Β. ... τη συμβολοσειρά `"0"`
   ◯ Γ. ... την τιμή της παράστασης `'3'-3`

6. Οι διαφορετικές μη αρνητικές τιμές που μπορεί να πάρει μία μεταβλητή `unsigned int` στη C σε σχέση με τις διαφορετικές μη αρνητικές τιμές που μπορεί να πάρει μία μεταβλητή `signed int` είναι ...
   ◯ Α. ... ίδιες σε πλήθος
   ◯ Β. ... περίπου διπλάσιου πλήθους
   ◯ Γ. ... ακριβώς διπλάσιου πλήθους

7. Η συνάρτηση `srand` της C ...
   ◯ Α. ... επιστρέφει ένα τυχαίο αριθμό
   ◯ Β. ... δίνει μία τυχαία τιμή σε κάποια θέση μνήμης
   ◯ Γ. ... αρχικοποιεί τη γεννήτρια τυχαίων αριθμών

8. Σε μία συνάρτηση C έχουμε σαν όρισμα δείκτη όταν θέλουμε να ...
   ◯ Α. ... περάσουμε στη συνάρτηση κάποια τιμή
   ◯ Β. ... μεταβάλει η συνάρτηση κάποια τιμή
   ◯ Γ. ... εκτυπωθεί από τη συνάρτηση ένα μήνυμα

9. Κατά την εκτέλεση ενός προγράμματος C, στη στοίβα φυλάσσονται οι ...
   ◯ Α. ... εξωτερικές μεταβλητές
   ◯ Β. ... αυτόματες μεταβλητές
   ◯ Γ. ... μεταβλητές για τις οποίες έχει δεσμευτεί χώρος δυναμικά

10. Αν σε μία συνάρτηση C έχει οριστεί το `char *str = "Hello";`, τότε πού φυλάσσεται η συμβολοσειρά `"Hello"`;
    ◯ Α. Στη στοίβα
    ◯ Β. Στο σωρό
    ◯ Γ. Στο χώρο του προγράμματος

11. Ο χώρος στον οποίο φυλάσσεται μία εξωτερική μεταβλητή της C ...
    ◯ Α. ... αποδεσμεύεται αυτόματα όταν η μεταβλητή πάψει να χρησιμοποιείται
    ◯ Β. ... αποδεσμεύεται με κλήση της συνάρτησης `free`
    ◯ Γ. ... παραμένει δεσμευμένος για όλη τη διάρκεια εκτέλεσης του προγράμματος

12. Όταν στη C μία συνάρτηση `f` ορίζεται σαν `void f(... ...`
    ◯ Α. ... έχει κενό σώμα
    ◯ Β. ... δεν δέχεται ορίσματα
    ◯ Γ. ... δεν επιστρέφει κάτι στο όνομά της

13. Η δήλωση `char str[] = "abc";` στη C είναι ισοδύναμη με την ...
    ◯ Α. ... `char str[3] = "abc";`
    ◯ Β. ... `char str[] = {'a', 'b', 'c'};`
    ◯ Γ. ... `char str[] = {'a', 'b', 'c', '\0'};`

14. Μία δομή επανάληψης `while` στη C ...
    ◯ Α. ... μπορεί να μην κάνει καμία επανάληψη
    ◯ Β. ... θα κάνει τουλάχιστον μία επανάληψη
    ◯ Γ. ... θα κάνει τουλάχιστον δύο επαναλήψεις

15. Πότε τα δύο πρωτότυπα συναρτήσεων `void f(int *p)` και `void f(int p[])` είναι ισοδύναμα στη C;
    ◯ Α. Πάντοτε
    ◯ Β. Όταν το `p` είναι δυναμικά δεσμευμένος πίνακας ακέραιων
    ◯ Γ. Όταν το `p` είναι στατικά δεσμευμένος πίνακας ακέραιων

16. Τι θα εκτυπώσει το παρακάτω τμήμα κώδικα C;

```
int x = 5, y = 9, z;
z = x++  ;  y = ++z - x++  ;  x = y-- - --z  ;
printf("%d %d %d\n", x, y, z);
```

    ◯ Α. `-7 -1 6`
    ◯ Β. `-7 -2 5`
    ◯ Γ. `-5 -1 5`

17. Τι θα εκτυπώσει το παρακάτω τμήμα κώδικα C;

```
int *p, x[] = {5, 7, 2, 8, 3};
p = x;
*p++ = x[3];
x[4] = (*p)++;
*x = ++(*p);
x[2] = *--p;
printf("%d %d %d %d %d %d\n", x[0], x[1], x[2], x[3], x[4], *(p+3));
```

    ◯ Α. `2 8 2 8 7 7`
    ◯ Β. `9 9 9 8 7 8`
    ◯ Γ. Τίποτα από τα προηγούμενα δύο

18. Τι θα εκτυπωθεί από την εντολή `./myargv good luck` αν το εκτελέσιμο πρόγραμμα `myargv` έχει προκύψει από το παρακάτω πρόγραμμα C;

```
int main (int argc, char **argv)
{  while (argc-- > 0)
      printf("%s ", *argv++);
   printf("\n");
   return 0; }
```

    ◯ Α. `good luck`
    ◯ Β. `./myargv good luck`
    ◯ Γ. `luck good`

19. Τι θα εκτυπώσει το παρακάτω πρόγραμμα C;

```
void swap(int x, int y)
{  int temp;
   temp = x  ;  x = y  ;  y = temp  ;}

int main(void)
{  int a = 12, b = 18;
   swap(a, b);
   printf("%d %d\n", a, b);
   return 0; }
```

◯ A. 18 12
◯ B. 12 18
◯ Γ. Τίποτα από τα προηγούμενα δύο

20. Τι θα εκτυπώσει το παρακάτω πρόγραμμα C;

```
void f(int x)
{ if (!x) return;
  f(x-1);
  printf("%d ", x); }

int main(void)
{  f(9);
   printf("\n");
   return 0; }
```

◯ A. ... 1 2 3 4 5 6 7 8 9
◯ B. ... 0 1 2 3 4 5 6 7 8
◯ Γ. ... Τίποτα από τα προηγούμενα δύο

21. Οι εντολές `*p = x;` και `p = &x;` ...
◯ A. ... είναι πάντοτε ταυτόσημες
◯ B. ... είναι ταυτόσημες υπό προϋποθέσεις
◯ Γ. ... δεν είναι ταυτόσημες

22. Όταν η συνάρτηση `malloc` της C αποτύχει, τι επιστρέφει;
◯ A. `EOF`
◯ B. `NULL`
◯ Γ. Τίποτα από τα προηγούμενα δύο

23. Έστω ότι καλούμε `fib(4)`, όπου `fib` είναι μία αναδρομική συνάρτηση C που έχει οριστεί ως εξής:

```
int fib(int n)
{  if (n == 0 || n == 1)
      return 1;
   else
      return (fib(n-1) + fib(n-2)); }
```

Πόσες φορές θα κληθεί η `fib(1)`;
◯ A. 1
◯ B. 2
◯ Γ. 3

24. Για να κατασκευάσουμε ένα ταξινομημένο δυαδικό δέντρο που να έχει ως κόμβους δεδομένους ακεραίους, ο κόμβος ρίζα του δέντρου ...
◯ A. ... πρέπει να είναι υποχρεωτικά ο ελάχιστος από τους ακεραίους
◯ B. ... πρέπει να είναι υποχρεωτικά ο ακέραιος εκείνος που είναι πλησιέστερος στη μέση τιμή των ακεραίων
◯ Γ. ... μπορεί να είναι οποιοσδήποτε από τους ακεραίους

25. Ποια είναι η πολυπλοκότητα του αλγορίθμου δυαδικής αναζήτησης σε πίνακα με n στοιχεία;
◯ A. $O(n)$
◯ B. $O(logn)$
◯ Γ. $O(n \cdot logn)$

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Γράψτε πρόγραμμα C το οποίο θα δέχεται στη γραμμή εντολής έναν ακέραιο N μεγαλύτερο του 1 και θα εκτυπώνει στην έξοδο όλους τους *άφθονους* (abundant) αριθμούς που είναι μικρότεροι ή ίσοι του N. Ένας αριθμός είναι άφθονος όταν το άθροισμα των διαιρετών του, εξαιρουμένου του εαυτού του, είναι μεγαλύτερο του αριθμού. Ο αλγόριθμος που θα χρησιμοποιήσετε πρέπει να έχει πολυπλοκότητα αυστηρά καλύτερη από $O(N^2)$. Μία ενδεικτική εκτέλεση είναι η εξής:

```
$ abundant 100
12 18 20 24 30 36 40 42 48 54 56 60 66 70 72 78 80 84 88 90 96 100
```

Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Αν στο πρόγραμμα που φαίνεται στη συνέχεια δοθεί σαν είσοδος ο αριθμός που σχηματίζεται από τα 9 τελευταία ψηφία του αριθμού μητρώου σας, τι θα εκτυπωθεί στην έξοδο;[1] Δικαιολογήστε την απάντησή σας.

```
#include <stdio.h>

int main(void)
{
    int x, y, z, u, t;
    double w;
    scanf("%d", &x);
    y = (x << 2) - (x & 0x0FFFFFFF);
    z = (x << 1) + (x ^ x);
    u = (y + z) / 5;
    t = (((u / 10000) | 01) - 1) / 10;
    w = (t / 100.0) - (t / 100) + (u % 1000);
    printf("%6.2f\n", w);
    return 0;
}
```

[1] Αν για οποιοδήποτε λόγο δεν έχετε αριθμό μητρώου προπτυχιακού φοιτητή του τμήματος, χρησιμοποιήστε τον 123405678.

Θέμα 4 (20/100): Εφ' όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Υπάρχουν θετικοί ακέραιοι για τους οποίους ισχύει ότι αν δοθούν σαν είσοδος στο πρόγραμμα που φαίνεται στη συνέχεια αριστερά, αυτό δεν τερματίζει. Αιτιολογήστε το προηγούμενο, δίνοντας ένα παράδειγμα θετικού ακεραίου για τον οποίο το πρόγραμμα δεν τερματίζει. Συμπληρώστε, στη συνέχεια δεξιά, την επαναδιατύπωση της συνάρτησης `main()` του προγράμματος, χωρίς να αλλάξετε τη συνάρτηση `next()`, έτσι ώστε το νέο πρόγραμμα να συμπεριφέρεται ακριβώς με τον ίδιο τρόπο για εισόδους `X` που το αρχικό τερμάτιζε, δηλαδή να εκτυπώνει "`X is happy`", αλλά για εισόδους που το αρχικό δεν τερμάτιζε, το νέο να τερματίζει και να εκτυπώνει "`X is unhappy`".

```c
#include <stdio.h>

int next(int x)
{
    int sum = 0;
    while (x) {
        int digit = x % 10;
        sum += digit * digit;
        x /= 10;
    }
    return sum;
}

int main(void)
{
    int x;
    scanf("%d", &x);
    printf("%d is ", x);
    do {
        x = next(x);
    } while (x != 1);
    printf("happy\n");
    return 0;
}
```

```c
#include <stdio.h>

int next(int x)
{
    int sum = 0;
    while (x) {
        int digit = x % 10;
        sum += digit * digit;
        x /= 10;
    }
    return sum;
}

int main(void)
{
    int x;


    scanf("%d", &x);
    printf("%d is ", x);









    if (              )
        printf("happy\n");
    else
        printf("unhappy\n");
    return 0;
}
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2016 (1/2/2016)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Ο προεπεξεργαστής της C παράγει ...
   - ◯ Α. ... αρχεία κώδικα C
   - ◯ Β. ... αντικειμενικά αρχεία
   - ◯ Γ. ... εκτελέσιμα αρχεία

2. Ο αριθμός 215 στο δεκαδικό σύστημα γράφεται ως ...
   - ◯ Α. ... 3113 στο οκταδικό σύστημα και ως 137 στο δεκαεξαδικό
   - ◯ Β. ... 327 στο οκταδικό σύστημα και ως 137 στο δεκαεξαδικό
   - ◯ Γ. ... 327 στο οκταδικό σύστημα και ως D7 στο δεκαεξαδικό

3. Ποια είναι η τιμή της αριθμητικής παράστασης `32 / 5 * 5` στη C;
   - ◯ Α. 1.28
   - ◯ Β. 32
   - ◯ Γ. 30

4. Αν η x είναι ακέραια μεταβλητή, ποια τιμή θα πάρει μετά την εκτέλεση της εντολής `x = ((y = 5) == 0);`
   - ◯ Α. 0
   - ◯ Β. 1
   - ◯ Γ. 5

5. Αν η x είναι τύπου `unsigned char`, ποια τιμή θα πάρει μετά την εντολή `x = ((15 << 2) | 0xEB) ^ 047;`
   - ◯ Α. 208
   - ◯ Β. 203
   - ◯ Γ. 216

6. Αν είχαμε καλέσει τη συνάρτηση `time` ακριβώς στην έναρξη της παρούσα εξέτασης, θα μας επέστρεφε την τιμή 1454309100. Τι θα μας επιστρέψει αν την καλέσουμε ακριβώς στη λήξη της εξέτασης;
   - ◯ Α. 1454309103
   - ◯ Β. 1454309280
   - ◯ Γ. 1454319900

7. Πού φυλάσσονται οι τιμές των μεταβλητών που ορίζονται εντός των συναρτήσεων και δεν έχουν δηλωθεί ως `static`;
   - ◯ Α. Στη στοίβα
   - ◯ Β. Στον σωρό
   - ◯ Γ. Στον εξωτερικό χώρο

8. Τι εκτυπώνεται από την παρακάτω εντολή;

```
printf("%c\n", putchar('A'+1)+2);
```

   - ◯ Α. B
   - ◯ Β. DB
   - ◯ Γ. BD

9. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int i = 0;
while (++i);
printf("*");
```

   - ◯ Α. Σχεδόν ακαριαία, ένα *
   - ◯ Β. Μετά από κάποια δευτερόλεπτα εκτέλεσης, ένα *
   - ◯ Γ. Θα εκτυπώνει συνεχώς *, επ΄ άπειρον

10. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
#define SUM 7+3
printf("%d %d\n", SUM*SUM, 12-SUM);
```

◯ A. 100 2
◯ B. 31 8
◯ Γ. Τίποτα, γιατί το παραπάνω τμήμα κώδικα έχει συντακτικό λάθος και δεν θα μεταγλωττισθεί

11. Η εντολή `y = (x++)++;` στη C είναι ...
◯ A. ... ισοδύναμη με τις εντολές `y = x+2; x++; x++;`
◯ B. ... ισοδύναμη με τις εντολές `y = x++; y = x++;`
◯ Γ. ... λάθος συντακτικά

12. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = 5;
while (x--);
printf("%d", x);
```

◯ A. 543210
◯ B. 0
◯ Γ. -1

13. Αν η μεταβλητή pc είναι δηλωμένη ως `char *` και η pd ως `double *`, τότε ...
◯ A. ... για την pc δεσμεύεται λιγότερος χώρος στη μνήμη απ' ότι για την pd
◯ B. ... δεσμεύεται ίδιου μεγέθους χώρος και για τις δύο μεταβλητές
◯ Γ. ... εξαρτάται από το σύστημα πόσος χώρος δεσμεύεται στη μνήμη για κάθε μεταβλητή

14. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
signed char c = 0xFF;
printf("%d\n", c);
```

◯ A. 255
◯ B. 1
◯ Γ. -1

15. Με τη δήλωση `int (*f)(void);` στη C ορίζεται το f ως ...
◯ A. ... συνάρτηση χωρίς ορίσματα που επιστρέφει δείκτη σε `int`
◯ B. ... μεταβλητή που είναι δείκτης σε συνάρτηση χωρίς ορίσματα που επιστρέφει δείκτη σε `int`
◯ Γ. ... μεταβλητή που είναι δείκτης σε συνάρτηση χωρίς ορίσματα που επιστρέφει `int`

16. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = 8, y = 12, z, w;
z = (x++) * (--y);
w = (z--) / (++x);
printf("%d %d %d %d\n", x, y, z, w);
```

◯ A. 10 11 99 11
◯ B. 10 11 87 8
◯ Γ. 10 11 96 11

17. Τι θα εκτυπωθεί από την εντολή `./myargv good luck` αν το εκτελέσιμο πρόγραμμα myargv έχει προκύψει από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
int main(int argc, char **argv)
{  while (--argc)
      printf("%c ", ***++argv);
   printf("\n");
   return 0; }
```

◯ A. good luck
◯ B. ./myargv good luck
◯ Γ. g l

18. Τι θα εκτυπωθεί από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
void g(int);
void f(int x)
{  if (!x) return;
   g(x-1); printf("f%d ", x); }
void g(int x)
{  if (!x) return;
   f(x-1); printf("g%d ", x); }
int main(void)
{  f(6); printf("\n"); return 0; }
```

◯ A. `g0 f1 g2 f3 g4 f5`
◯ B. `f6 g5 f4 g3 f2 g1`
◯ Γ. `g1 f2 g3 f4 g5 f6`

19. Η εντολή `&x = p;` στη C ...
◯ A. ... αναθέτει στον δείκτη `p` τη διεύθυνση της μεταβλητής `x`
◯ B. ... αναθέτει στη μεταβλητή `x` το περιεχόμενο της θέσης μνήμης που δείχνει ο δείκτης `p`
◯ Γ. ... είναι συντακτικά λανθασμένη

20. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
char *str1 = "mystring", str2[10];
strcpy(str2, str1);
printf("%sequal\n", str1 == str2 ? "" : "not ");
```

◯ A. `not equal`
◯ B. `equal`
◯ Γ. Τίποτα από τα προηγούμενα δύο, γιατί θα προκύψει σφάλμα εκτέλεσης

21. Με ποια έκφραση είναι ισοδύναμη η `p->value` στη C;
◯ A. `*p.value`
◯ B. `(*p).value`
◯ Γ. `*(p.value)`

22. Το πλήθος των στοιχείων μίας απλά συνδεδεμένης λίστας ...
◯ A. ... πρέπει να δηλωθεί στην αρχή του προγράμματος
◯ B. ... καθορίζεται δυναμικά με τη συνάρτηση `malloc` και στη συνέχεια παραμένει σταθερό
◯ Γ. ... μπορεί να μεταβάλλεται δυναμικά κατά την εκτέλεση του προγράμματος

23. Ποια πιστεύετε ότι θα ήταν μία κατάλληλη εναλλακτική ονομασία για τα ταξινομημένα δυαδικά δέντρα;
◯ A. ισοζυγισμένα δέντρα
◯ B. οπωροφόρα δέντρα
◯ Γ. δέντρα αναζήτησης

24. Η μέθοδος ταξινόμησης του σωρού ...
◯ A. ... είναι αναδρομική
◯ B. ... έχει πολυπλοκότητα $O(n \cdot logn)$, για πίνακα με n στοιχεία
◯ Γ. ... απαιτεί δυναμικά δεσμευμένο πίνακα για να λειτουργήσει σωστά

25. Ποια είναι η πολυπλοκότητα χρόνου του καλύτερου αλγορίθμου που μπορούμε να γράψουμε για να βρούμε το άθροισμα των διαιρετών ενός ακεραίου αριθμού N;
◯ A. $O(N^2)$
◯ B. $O(N)$
◯ Γ. $O(\sqrt{N})$

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Γράψτε πρόγραμμα C το οποίο θα δέχεται στη γραμμή εντολής έναν ακέραιο N μεγαλύτερο του 1 και θα εκτυπώνει στην έξοδο όλους τους *πεμπτοδύναμους* αριθμούς που είναι μικρότεροι ή ίσοι του N. Ένας ακέραιος αριθμός μεγαλύτερος του 1 είναι πεμπτοδύναμος όταν το άθροισμα των πέμπτων δυνάμεων των ψηφίων του ισούται με τον αριθμό. Για παράδειγμα, ο αριθμός 4150 είναι πεμπτοδύναμος, αφού $4^5 + 1^5 + 5^5 + 0^5 = 4150$. Μία ενδεικτική εκτέλεση είναι η εξής:

```
$ ./fifthpow 354294
4150 4151 54748 92727 93084 194979
```

Δεν επιτρέπεται η χρήση πινάκων (συμπεριλαμβανομένων και των συμβολοσειρών).
Παρατηρήστε ότι στην ενδεικτική εκτέλεση δώσαμε σαν N το 354294. Αποδείξτε μαθηματικά ότι δεν μπορεί να υπάρξει πεμπτοδύναμος αριθμός μεγαλύτερος από 354294 (bonus ερώτηση).

Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

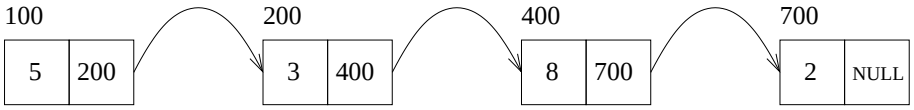
Θεωρήστε ότι ένας κόμβος λίστας ακεραίων ορίζεται, κατά τα γνωστά, ως:

```
typedef struct listnode *Listptr;
struct listnode {
  int value;
  Listptr next; };
```

Η ελλιπής συνάρτηση που δίνεται στη συνέχεια δέχεται ως όρισμα μία λίστα ακεραίων και επιστρέφει στο όνομά της μία λίστα με αντεστραμμένη τη σειρά των κόμβων της αρχικής. Η συνάρτηση αναδιατάσσει τους υπάρχοντες κόμβους, δηλαδή δεν δημιουργεί νέους, και η αναδιάταξη επιτυγχάνεται με κατάλληλες τροποποιήσεις στους δείκτες που περιέχονται στους κόμβους, χωρίς να μεταβάλλονται τα δεδομένα (ακέραιοι) των κόμβων. Τέλος, η συνάρτηση έχει πολυπλοκότητα $O(N)$, όπου N είναι το πλήθος των στοιχείων της λίστας. Συμπληρώστε τις θέσεις με τις τελείες (όχι κατ΄ ανάγκη όλες, αλλά, σίγουρα, όχι περισσότερες από όσες δίνονται) ώστε η συνάρτηση να λειτουργεί με τον τρόπο που περιγράφηκε προηγουμένως. Δεν επιτρέπεται να χρησιμοποιήσετε άλλες μεταβλητές, πέραν αυτών που ορίζονται ήδη στη συνάρτηση.

```
Listptr revlist(Listptr list)
{  Listptr newlist ....., tmplist;
   ...........................
   while (.....................) {
      ........................
      ........................
      ........................
      ........................ }
   return ..................... ; }
```

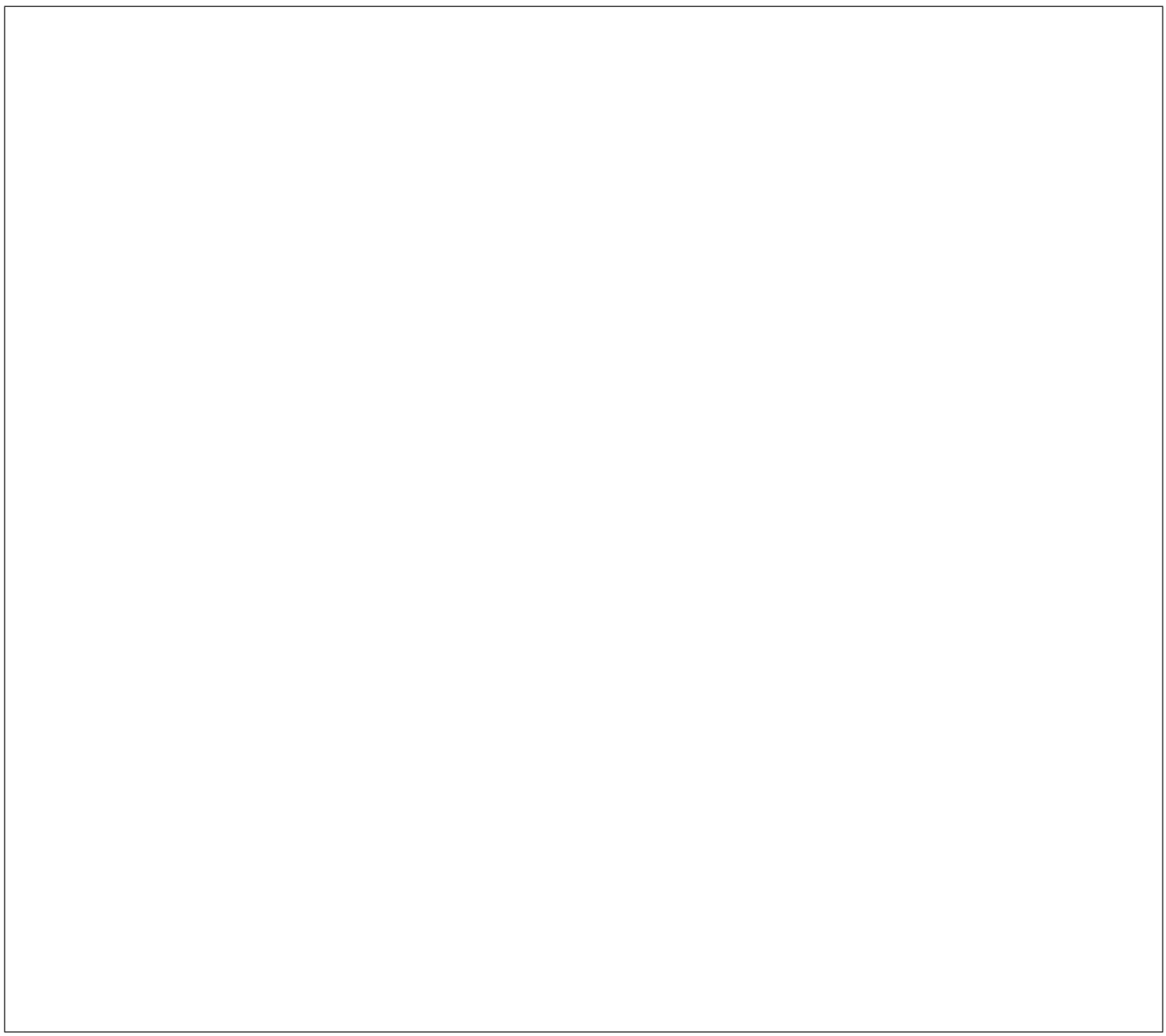
Έστω η παρακάτω λίστα (οι αριθμοί έξω από τους κόμβους είναι οι διευθύνσεις τους). Με τι τιμή πρέπει να κληθεί η συνάρτηση `revlist` για να αντιστρέψει τη λίστα αυτή και ποια τιμή θα επιστρέψει στο όνομά της; Στο πλαίσιο που ακολουθεί, δείξτε πώς μεταβάλλονται από τη συνάρτηση σταδιακά οι τιμές των μεταβλητών της, όταν αυτή κληθεί για να αντιστρέψει τη συγκεκριμένη λίστα, και δείξτε σχηματικά την τελική λίστα.



Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Γράψτε ένα πρόγραμμα C που να υπολογίζει όλα τα 2^{20} *mod* 3, 2^{200} *mod* 9, 2^{2000} *mod* 27, 2^{20000} *mod* 81, ..., $2^{2000000000}$ *mod* 19683, δηλαδή όλα τα $2^{2 \cdot 10^n}$ *mod* 3^n (υπόλοιπο διαίρεσης με το 3^n του 2 υψωμένου στη δύναμη 2 επί 10^n), για κάθε $n = 1, 2, 3, 4, \ldots, 9$. Παράδειγμα εκτέλεσης:

```
$ time ./compute_2_exp_2_times_10_exp_n_mod_3_exp_n
n = 1, mod = 1
n = 2, mod = 4
n = 3, mod = 4
n = 4, mod = 31
n = 5, mod = 139
n = 6, mod = 166
n = 7, mod = 1165
n = 8, mod = 2407
n = 9, mod = 11182
0.000u 0.000s 0:00.00 0.0%      0+0k 0+0io 0pf+0w
```



ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2006 (21/1/2006)

1. (α΄) Σχολιάστε, *σε 10 το πολύ γραμμές*, τη χρήση της αναδρομής στον προγραμματισμό. Αναφέρετε πλεονεκτήματα και μειονεκτήματά της, πιθανά προβλήματα που μπορεί να παρουσιαστούν, σημεία που πρέπει να προσεχθούν κατά τη χρήση της, ή ό,τι άλλο σχετικό κρίνετε ενδιαφέρον.

   (β΄) Γράψτε αναδρομική συνάρτηση C, έστω την `int recseq(int n)`, που, για δεδομένο `n`, να υπολογίζει και να επιστρέφει στο όνομά της τον όρο a_n της ακολουθίας που ορίζεται ως εξής:

$$a_n = \begin{cases} 1 & \text{αν } 0 \le n \le 4 \\ a_{\lfloor n/2 \rfloor} + a_{\lfloor 2n/3 \rfloor} + a_{n-5} & \text{αν } n \ge 5 \end{cases}$$

   Δεν χρειάζεται να γράψετε `main` συνάρτηση που να καλεί την `recseq`. Το $\lfloor n/2 \rfloor$ είναι το ακέραιο μέρος του $n/2$, ή, αλλιώς, το πηλίκο της διαίρεσης του n διά 2. Ομοίως και για το $\lfloor 2n/3 \rfloor$.

   (γ΄) Περιγράψτε, *σε 5 το πολύ γραμμές*, πώς θα υλοποιούσατε τη συνάρτηση `recseq` με επαναληπτικό τρόπο, χωρίς αναδρομή. Δεν σας ζητείται να γράψετε πρόγραμμα, απλώς να δώσετε μία στοιχειώδη περιγραφή της υλοποίησης.

2. Έστω το παρακάτω C πρόγραμμα:

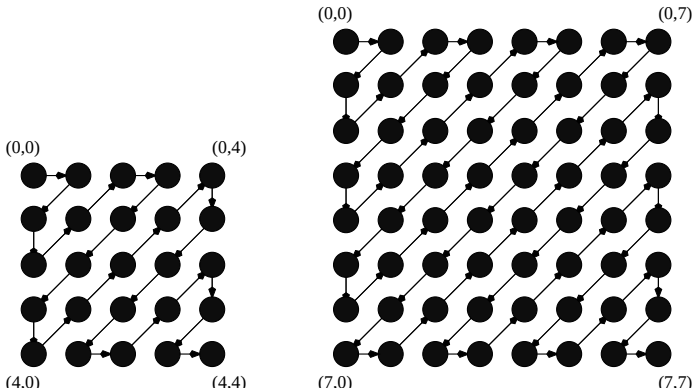
```
#include <stdio.h>

void f(int a, int *q, int **r)
{  printf("%d %d\n", a++, *q++);
   printf("%d %d %d\n", a++, (*q)++, *(*r)++); }

main()
{  int x[] = {10, 20, 30, 40, 50, 60, 70};
   int i, *p;
   for (i=0, p=x ; i<3 ; i++, p++)
      f(*p, p, &p);
   for ( ; p>=x ; p--)
      printf("%d ", *p);
   printf("\n"); }
```

   Τι θα εκτυπωθεί από την εκτέλεση αυτού του προγράμματος; Δικαιολογήστε στοιχειωδώς την απάντησή σας, *σε μισή σελίδα το πολύ*.

3. Εικάζεται, αλλά δεν έχει αποδειχθεί μαθηματικά, ότι αν ξεκινήσουμε από ένα θετικό ακέραιο αριθμό n και πάρουμε σαν επόμενό του τον $n/2$, αν ο n είναι άρτιος, ή τον $3n+1$, αν ο n είναι περιττός, συνεχίζοντας με αυτόν τον τρόπο, κάποια στιγμή θα καταλήξουμε στο 1. Για παράδειγμα, αν ξεκινήσουμε από το $n = 22$, η ακολουθία αριθμών που θα πάρουμε με αυτή τη διαδικασία θα είναι οι 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. Για κάθε αριθμό n, το *μήκος κύκλου* του είναι το πλήθος των αριθμών που περιλαμβάνονται στην ακολουθία που δημιουργείται με τον παραπάνω τρόπο, συμπεριλαμβανομένου του n και του 1. Για παράδειγμα, το μήκος κύκλου του 22 είναι 16. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`conjecture.c`"), το οποίο να δέχεται στη γραμμή εντολής δύο θετικούς ακεραίους `k` και `m` και να βρίσκει εκείνο τον αριθμό `n` από όλους τους αριθμούς μεταξύ `k` και `m`, συμπεριλαμβανομένων, που έχει το μεγαλύτερο μήκος κύκλου. Επίσης, το πρόγραμμά σας να εκτυπώνει την ακολουθία αριθμών που προκύπτει από τον αριθμό με μέγιστο μήκος κύκλου, δηλαδή τον n. Αν υπάρχουν δύο ή περισσότεροι αριθμοί με το ίδιο μέγιστο μήκος κύκλου, σαν απάντηση να δίνεται ο μικρότερος απ΄ αυτούς. Παραδείγματα εκτέλεσης είναι τα εξής:

```
% ./conjecture 20 24
n = 22, cycle length = 16
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
% ./conjecture 1 20
n = 18, cycle length = 21
18 9 28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
% ./conjecture 20 30
n = 27, cycle length = 112
27 82 41 124 62 31 94 47 142 71 214 107 322 161 484 242 .......
%
```

4. Παρατηρήστε τη σειρά επίσκεψης των κόμβων στα παρακάτω τετραγωνικά πλέγματα, διαστάσεων 5×5 και 8×8, αντίστοιχα. Η σειρά επίσκεψης είναι εκείνη που δείχνουν τα βέλη, ξεκινώντας από το στοιχείο $(0,0)$, επάνω αριστερά, πηγαίνοντας μετά δεξιά στο στοιχείο $(0,1)$, μετά διαγώνια κάτω αριστερά στο $(1,0)$, μετά κάτω στο $(2,0)$, μετά διαγώνια επάνω δεξιά και ούτω καθεξής, μέχρι να φτάσουμε στο τελευταίο στοιχείο του πλέγματος, κάτω δεξιά, το $(4,4)$ ή το $(7,7)$, αντίστοιχα για τα δύο πλέγματα. Η διαδικασία αυτή αποτελεί μέρος της μεθοδολογίας συμπίεσης εικόνας σε μορφή jpeg.



Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`jpeg.c`"), το οποίο να παίρνει από τη γραμμή εντολής τη διάσταση n ενός τετραγωνικού πλέγματος $n \times n$, με στοιχεία ακεραίους, να διαβάζει από την είσοδο τα δεδομένα του πλέγματος, δηλαδή τους ακεραίους στους κόμβους του, κατά γραμμές και να εκτυπώνει τα δεδομένα αυτά με τη zigzag σειρά του σχήματος. Για παράδειγμα:

```
% ./jpeg 5
234   4  56 178 222
  1  23 124  45 100
 24   2   9   0  82
156  77  99  83   7
  8 100  87  12  34
234 4 1 24 23 56 178 124 2 156 8 77 9 45 222 100 0 99 100 87 83 82 7 12 34
% cat jpeg.txt
122  87  34 100   1  34 27 89
239   3 120  23 156 126  7  0
 37  22   4   0 111  99 23 76
 23  10  10  10  76 100 98  7
121 234 255  88 126 128 64 32
 24  23   4  77  42   1  8 33
  8  56  21   5  88 124 78  9
222 223 178  66 236  12 87  3
% ./jpeg 8 < jpeg.txt
122 87 239 37 3 34 100 120 22 23 121 10 4 23 1 34 156 0 10 234 24 8 .......
%
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2010 (23/1/2010)

1. (α΄) Ποια είναι η σχέση των πινάκων και των συμβολοσειρών στη γλώσσα προγραμματισμού C;

   (β΄) Γιατί αρκετά συχνά χρειαζόμαστε τη συνάρτηση `atoi` όταν επεξεργαζόμαστε τα ορίσματα της γραμμής εντολών σ΄ ένα πρόγραμμα C;

   (γ΄) Είναι δυνατόν σ΄ ένα πρόγραμμα C μία συμβολοσειρά να αποθηκεύεται στη στοίβα εκτέλεσης; Στο σωρό; Στον εξωτερικό χώρο των στατικών/καθολικών δεδομένων; Κάπου αλλού, ίσως; Γράψτε ένα πρόγραμμα C, στο οποίο οι συμβολοσειρές `"instack"`, `"inheap"`, `"instatic"` και `"elsewhere"` να αποθηκεύονται, αντίστοιχα, στους χώρους που αναφέρθηκαν προηγουμένως, όπου είναι εφικτό.

   (δ΄) Υπάρχει περίπτωση κάποια από τις συμβολοσειρές του προηγούμενου ερωτήματος να μην είναι επιτρεπτό να τροποποιηθεί; Αν ναι, προσθέστε και κάποια εντολή στο πρόγραμμα που γράψατε, η οποία να είναι σε θέση να προκαλέσει λάθος κατά την εκτέλεση του προγράμματος, λόγω του προηγούμενου περιορισμού.

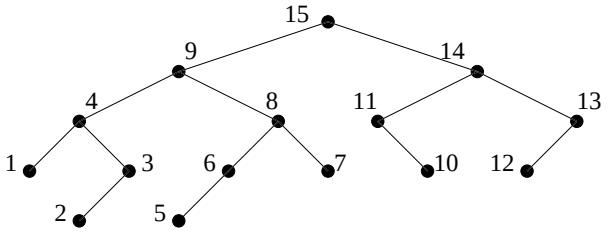
   (ε΄) Στην πρότυπη βιβλιοθήκη της C περιλαμβάνεται και η συνάρτηση

   ```
   char *strrchr(const char *s, int c)
   ```

   Η συνάρτηση αυτή επιστρέφει ένα δείκτη στην **τελευταία** εμφάνιση του χαρακτήρα που έχει ASCII κωδικό `c` μέσα στη συμβολοσειρά `s`. Αν δεν υπάρχει τέτοιος χαρακτήρας στην `s`, επιστρέφει `NULL`. Δώστε μία πιθανή υλοποίηση της συνάρτησης αυτής, **χωρίς να χρησιμοποιήσετε άλλες συναρτήσεις της πρότυπης βιβλιοθήκης**.

2. Στην C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου ακεραίων ως εξής:

```
typedef struct treenode *Treeptr;

struct treenode {
  int value;
  Treeptr left;
  Treeptr right;
};
```

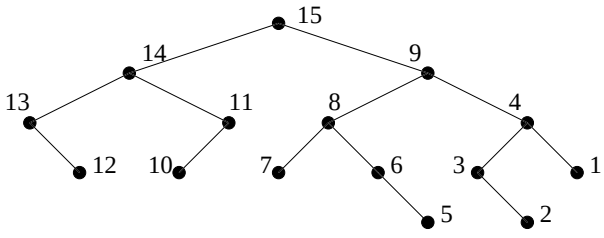


Σχήμα 1

   (α΄) Στο Σχήμα 1, οι αριθμοί που φαίνονται στους κόμβους του δυαδικού δέντρου υποδεικνύουν τη σειρά επίσκεψης των κόμβων κατά τη λεγόμενη *μεταδιατεταγμένη διάσχιση* (postorder traversal). Προσπαθήστε να κατανοήσετε τη λογική της διάσχισης αυτής και υλοποιήστε μία συνάρτηση C, με πρωτότυπο `void print_postorder(Treeptr)`, η οποία να παίρνει σαν όρισμα ένα δυαδικό δέντρο ακεραίων και να εκτυπώνει τα περιεχόμενα των κόμβων του σύμφωνα με τη μεταδιατεταγμένη διάσχιση. Για παράδειγμα, αν οι αριθμοί που φαίνονται στο παραπάνω δέντρο ήταν και τα αντίστοιχα περιεχόμενα των κόμβων, η συνάρτηση θα έπρεπε να εκτυπώσει:

   ```
   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
   ```

   (β΄) Υλοποιήστε μία συνάρτηση C, με πρωτότυπο `void mirror_tree(Treeptr)`, η οποία να μετασχηματίζει ένα δυαδικό δέντρο ακεραίων στο "κατοπτρικό" του. Δηλαδή, σε κάθε κόμβο του αρχικού, το αριστερό του παιδί να γίνεται δεξί και το δεξί αριστερό. Στο Σχήμα 2 φαίνεται το κατοπτρικό δέντρο αυτού του Σχήματος 1. Με ποια σειρά θα εκτυπωνόντουσαν οι κόμβοι του δέντρου του Σχήματος 2, σύμφωνα με τη μεταδιατεταγμένη διάσχισή του;



Σχήμα 2

3. Η εικασία (μαθηματική πρόταση που δεν έχει αποδειχθεί, αλλά πιστεύεται ότι είναι σωστή) του Waring λέει ότι κάθε περιττός ακέραιος αριθμός μεγαλύτερος του 2 είτε είναι πρώτος, είτε μπορεί να γραφεί σαν άθροισμα τριών πρώτων αριθμών. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`waring.c`"), το οποίο να καλείται με δύο ακεραίους M και N σαν ορίσματα στη γραμμή εντολής και να εκτυπώνει, για κάθε **μη-πρώτο περιττό** που είναι μεγαλύτερος ή ίσος του M και μικρότερος ή ίσος του N όλους τους τρόπους με τους οποίους αυτός μπορεί να γραφεί σαν άθροισμα τριών πρώτων αριθμών. Εννοείται ότι δεν πρέπει να θεωρούνται σαν διαφορετικοί τρόποι γραφής οι εναλλακτικές διατάξεις των προσθετέων. Το `21 = 2 + 2 + 17` δεν διαφέρει από το `21 = 2 + 17 + 2`, προφανώς. Ένα παράδειγμα εκτέλεσης φαίνεται δεξιά.

```
% ./waring 17 32
21 = 2 + 2 + 17
21 = 3 + 5 + 13
21 = 3 + 7 + 11
21 = 5 + 5 + 11
21 = 7 + 7 + 7
25 = 3 + 3 + 19
25 = 3 + 5 + 17
25 = 3 + 11 + 11
25 = 5 + 7 + 13
25 = 7 + 7 + 11
27 = 2 + 2 + 23
27 = 3 + 5 + 19
27 = 3 + 7 + 17
27 = 3 + 11 + 13
27 = 5 + 5 + 17
27 = 5 + 11 + 11
27 = 7 + 7 + 13
```

4. Όπως γνωρίζετε από τη Γραμμική Άλγεβρα, η ορίζουσα ενός τετραγωνικού πίνακα είναι ένας χαρακτηριστικός αριθμός του πίνακα με ιδιαίτερη σημασία σε πολλές περιπτώσεις. Ο απλούστερος τρόπος να υπολογισθεί η ορίζουσα ενός τετραγωνικού πίνακα είναι μέσω του τύπου του Laplace. Σύμφωνα με αυτό τον τύπο, αναπτύσσουμε την ορίζουσα ως προς μία γραμμή ή μία στήλη. Έστω ότι εφαρμόζουμε τον τύπο για ανάπτυξη ως προς την πρώτη στήλη. Τότε, η ορίζουσα του πίνακα ισούται με το άθροισμα των γινομένων, με εναλλασσόμενα πρόσημα θετικά και αρνητικά, κάθε στοιχείου της στήλης επί την ορίζουσα του πίνακα που προκύπτει κάθε φορά αν διαγράψουμε από τον αρχικό πίνακα την πρώτη στήλη και τη γραμμή στην οποία βρίσκεται το εν λόγω στοιχείο της πρώτης στήλης. Για παράδειγμα:

$$\begin{vmatrix} 3 & 5 & -1 & 2 \\ 6 & 2 & 5 & 12 \\ 7 & 1 & 9 & -8 \\ -4 & 0 & 8 & 10 \end{vmatrix} = 3 \cdot \begin{vmatrix} 2 & 5 & 12 \\ 1 & 9 & -8 \\ 0 & 8 & 10 \end{vmatrix} - 6 \cdot \begin{vmatrix} 5 & -1 & 2 \\ 1 & 9 & -8 \\ 0 & 8 & 10 \end{vmatrix} + 7 \cdot \begin{vmatrix} 5 & -1 & 2 \\ 2 & 5 & 12 \\ 0 & 8 & 10 \end{vmatrix} - (-4) \cdot \begin{vmatrix} 5 & -1 & 2 \\ 2 & 5 & 12 \\ 1 & 9 & -8 \end{vmatrix}$$

Δηλαδή, ο υπολογισμός της ορίζουσας ενός $N \times N$ πίνακα ανάγεται στον υπολογισμό των οριζουσών N πινάκων $(N-1) \times (N-1)$. Η ορίζουσα ενός πίνακα 1×1 ισούται με το μοναδικό στοιχείο του πίνακα. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`determinant.c`"), το οποίο να καλείται με ένα θετικό ακέραιο N στη γραμμή εντολής και αφού διαβάσει από την πρότυπη είσοδο (stdin) τα στοιχεία ενός τετραγωνικού πίνακα ακεραίων $N \times N$, να υπολογίζει και να εκτυπώνει την ορίζουσά του. Ένα παράδειγμα εκτέλεσης:

```
% ./determinant 1
10
The determinant of the given matrix equals to 10
% ./determinant 2
4 7
3 8
The determinant of the given matrix equals to 11
% ./determinant 4
   3   5  -1   2
   6   2   5  12
   7   1   9  -8
  -4   0   8  10
The determinant of the given matrix equals to -7928
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2011 (24/1/2011)

1. (α΄) Τι θα εκτυπωθεί από το πρόγραμμα C στο διπλανό σχήμα; Γνωρίζετε αν σημαίνει κάτι το αποτέλεσμα;

```
#include <stdio.h>
int main(void)
{  int a;
   for (a = 1 ; a <= 10 ; a++) {
      printf("%2d - ", a);
      switch (a) {
         case 8:  printf("v");
         case 3:  printf("i");
                  break;
         case 1:  printf("i");
                  break;
         case 4:  printf("i");
         case 5:
         case 6:
         case 7:  printf("v");
                  break;
         case 9:  printf("i");
         case 10: printf("x");
                  break; }
      switch (a) {
         case 8:
         case 3:
         case 2:
         case 7:  printf("i");
         case 6:  printf("i"); }
      printf("\n"); }
      return 0; }
```

   (β΄) Πότε χρειάζεται η συνάρτηση `realloc` της C; Έχει κάποιο βασικό μειονέκτημα η χρήση της; Αν ναι, τι εναλλακτικό έχουμε;

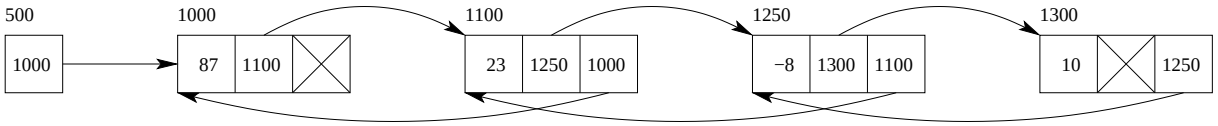
   (γ΄) Έστω το παρακάτω τμήμα προγράμματος C:

```
while (!feof(ifp)) {
  n = fread(buf, sizeof(char),
            sizeof(buf), ifp);
  fwrite(buf, sizeof(char), n, ofp); }
```

   Αν τα `ifp` και `ofp` είναι ρεύματα που αντιστοιχούν σε αρχεία που έχουν ανοίξει αμέσως πριν από αυτό το τμήμα προγράμματος, το πρώτο για διάβασμα και το δεύτερο για γράψιμο, και τα οποία κλείνουν αμέσως μετά, ποιος είναι ο στόχος αυτού του τμήματος προγράμματος; Τι διαφορά υπάρχει στην εκτέλεσή του, αν το `buf` έχει οριστεί σαν `char buf[4]` ή σαν `char buf[4096]`;

   (δ΄) Σχετικά με τις μεθόδους ταξινόμησης πινάκων, ποιο είναι το βασικό πλεονέκτημα *i)* της γρήγορης μεθόδου έναντι της μεθόδου της εισαγωγής, *ii)* της μεθόδου της συγχώνευσης έναντι της γρήγορης μεθόδου και *iii)* της μεθόδου του σωρού έναντι της μεθόδου της συγχώνευσης;

2. Εκτός από τις απλά συνδεδεμένες λίστες που γνωρίζετε ήδη, μπορούμε να χρησιμοποιήσουμε στον προγραμματισμό και διπλά συνδεδεμένες λίστες. Σε κάθε κόμβο μίας διπλά συνδεδεμένης λίστας, εκτός από ένα δείκτη στον επόμενό του κόμβο, υπάρχει και ένας δείκτης στον προηγούμενό του κόμβο. Ένα παράδειγμα διπλά συνδεδεμένης λίστας ακεραίων φαίνεται στο σχήμα.



   (α΄) Αφού ορίσετε τη δομή `struct` ενός κόμβου διπλά συνδεδεμένης λίστας ακεραίων, υλοποιήστε συνάρτηση C η οποία να εκτυπώνει τα περιεχόμενα των κόμβων πρώτα σε ευθεία φορά και μετά αντίστροφα. Μια ενδεικτική εκτύπωση της συνάρτησης για τη λίστα του σχήματος είναι η εξής:

```
87 --> 23 --> -8 --> 10 --> END --> 10 --> -8 --> 23 --> 87
```

   (β΄) Υλοποιήστε συνάρτηση C η οποία να εισάγει στην αρχή διπλά συνδεδεμένης λίστας ακεραίων ένα κόμβο με δεδομένο ακέραιο ως περιεχόμενο.

   (γ΄) Δώστε **μόνο** το πρωτότυπο συνάρτησης C που να διαγράφει το N-οστό στοιχείο μίας διπλά συνδεδεμένης λίστας ακεραίων. Με ποιες τιμές στα ορίσματά της θα έπρεπε να καλέσετε τη συνάρτησή

σας, ώστε να διαγράψει το 3ο στοιχείο της λίστας του σχήματος; Επίσης, σχεδιάστε τη λίστα μετά τη διαγραφή του 3ου στοιχείου της.

3. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`perfcube.c`"), το οποίο να καλείται με έναν ακέραιο N σαν όρισμα στη γραμμή εντολής και να εκτυπώνει για όλους του θετικούς ακεραίους που είναι μικρότεροι ή ίσοι του N, εκείνους για τους οποίους ο κύβος τους μπορεί να γραφεί σαν άθροισμα τριών κύβων ακεραίων, π.χ. $18^3 = 2^3 + 12^3 + 16^3$, αλλά και τον τρόπο γραφής τους. Αν ο κύβος ενός αριθμού μπορεί να γραφεί σαν άθροισμα τριών κύβων με περισσότερους από έναν τρόπο, το πρόγραμμα να εκτυπώνει όλους τους τρόπους. Φυσικά, δεν θεωρούνται διαφορετικοί τρόποι οι αναδιατάξεις των προσθετέων, π.χ. $18^3 = 16^3 + 2^3 + 12^3$. Μία ενδεικτική εκτέλεση είναι η εξής:

```
% ./perfcube 26
6^3 = 3^3 + 4^3 + 5^3
9^3 = 1^3 + 6^3 + 8^3
12^3 = 6^3 + 8^3 + 10^3
18^3 = 2^3 + 12^3 + 16^3
18^3 = 9^3 + 12^3 + 15^3
19^3 = 3^3 + 10^3 + 18^3
20^3 = 7^3 + 14^3 + 17^3
24^3 = 12^3 + 16^3 + 20^3
25^3 = 4^3 + 17^3 + 22^3
%
```

4. Στα κελιά ενός ορθογωνίου πλέγματος ζουν κάποιες οντότητες που μπορεί να είναι τριών ειδών, *πέτρα* (`R`/rock), *ψαλίδι* (`S`/scissors) ή *χαρτί* (`P`/paper). Η οντότητα `R` είναι ισχυρότερη της `S`, η `S` της `P` και η `P` της `R`. Κατά τη διάρκεια μίας ημέρας, οι γειτονικές οντότητες αλληλεπιδρούν, με αποτέλεσμα, στο τέλος της ημέρας, η κατάσταση του πλέγματος να μετασχηματίζεται ως εξής. Για κάθε οντότητα, αν σε κάποιο γειτονικό κελί της, οριζόντια ή κατακόρυφα, υπάρχει τουλάχιστον μία ισχυρότερη από αυτήν, τότε αυτή μετατρέπεται στο είδος της ισχυρότερης. Για παράδειγμα, αν κάποιο `S` έχει γειτονικό του τουλάχιστον ένα `R`, στο τέλος της ημέρας, θα γίνει και αυτό `R`. Ένα `P` μπορεί να γίνει `S`, αν έχει γείτονα `S` και ένα `R` μπορεί να γίνει `P`, αν έχει γείτονα `P`. Όλες οι μετατροπές γίνονται ακαριαία, στο τέλος της ημέρας, ταυτόχρονα για όλες τις οντότητες, ανάλογα με το αν η οντότητα έχει γείτονα ισχυρότερο από αυτήν ή όχι. Αν μία οντότητα δεν έχει ισχυρότερο γείτονα, δεν αλλάζει είδος.

   Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`rsp.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (stdin) τρεις ακεραίους N, M και D, να γεμίζει με τυχαίο τρόπο ένα πλέγμα $N \times M$ με οντότητες `R`, `S` και `P`, ισοπίθανα κατανεμημένες, και να δείχνει την εξέλιξη του πλέγματος σε διάστημα D ημερών. Δηλαδή, πρέπει να εκτυπώνονται οι καταστάσεις του πλέγματος για κάθε μία από τις D ημέρες και, επιπλέον, η τελική κατάσταση που προκύπτει μετά και την τελευταία ημέρα ($D+1$ καταστάσεις συνολικά). Μία ενδεικτική εκτέλεση φαίνεται στη συνέχεια. Μπορείτε να επιλέξετε να κάνετε τις εκτυπώσεις με κατακόρυφο τρόπο (κάθε κατάσταση κάτω από την προηγούμενή της) και όχι οριζόντια, όπως φαίνεται στην ενδεικτική εκτέλεση. Τι επιπτώσεις έχει στην υλοποίηση του προγράμματός σας το ποια μορφή εκτύπωσης θα επιλέξετε;

```
% ./rsp
3 4 6
Day 1      Day 2      Day 3      Day 4      Day 5      Day 6
RRSS ----> RRRS ----> PRRR ----> SPRR ----> RSPR ----> RRSP ----> PRRS
RSSS       PRRS       SPRR       RSPR       RRSP       PRRS       SPRR
PSRS       SRRR       RRRR       RPRR       PSPR       SRSP       RRRS
%
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2013 (28/1/2013)

1. (α΄) Ζητήθηκε από κάποιον να ορίσει μία μακροεντολή `CUBE(X)` για τον προεπεξεργαστή της C, η οποία να υλοποιεί την ύψωση στον κύβο ενός πραγματικού αριθμού και εκείνος απάντησε ως εξής:

   ```
   #define CUBE(X) (X)*(X)*(X)
   ```

   Συμφωνείτε με την απάντηση; Αν όχι, δώστε ένα παράδειγμα χρήσης της μακροεντολής που δεν θα οδηγούσε στο αναμενόμενο αποτέλεσμα και προτείνετε τον, κατά τη γνώμη σας, σωστό ορισμό, ο οποίος θα χειρίζεται σωστά το παράδειγμα που δώσατε.

   (β΄) Στην πρότυπη βιβλιοθήκη της C περιλαμβάνεται και η συνάρτηση

   ```
   char *strstr(const char *s1, const char *s2)
   ```

   Η συνάρτηση αυτή αναζητά την πρώτη εμφάνιση της συμβολοσειράς `s2` μέσα στη συμβολοσειρά `s1` και επιστρέφει δείκτη στον πρώτο χαρακτήρα της ευρεθείσας εμφάνισης της `s2` μέσα στην `s1`, δηλαδή την υποσυμβολοσειρά της `s1` από τον χαρακτήρα αυτό μέχρι το τέλος της. Αν δεν υπάρχει εμφάνιση της `s2` μέσα στην `s1`, η συνάρτηση επιστρέφει `NULL`. Αν η `s2` είναι η κενή συμβολοσειρά (`""`), η συνάρτηση επιστρέφει την `s1`. Για παράδειγμα, έστω ότι `s1="abcdecdfgcdfh"`. Αν `s2="cdf"`, τότε η συνάρτηση θα επιστρέψει `"cdfgcdfh"`. Αν `s2="cdef"`, τότε η συνάρτηση θα επιστρέψει `NULL`. Αν `s2=""`, τότε η συνάρτηση θα επιστρέψει `"abcdecdfgcdfh"`. Δώστε μία πιθανή υλοποίηση της συνάρτησης αυτής, **χωρίς να χρησιμοποιήσετε άλλες συναρτήσεις της πρότυπης βιβλιοθήκης**.

   (γ΄) Εξηγήστε γιατί οι παρακάτω συναρτήσεις `ds1` και `ds2` επιστρέφουν το ίδιο αποτέλεσμα όταν κληθούν με τον ίδιο ακέραιο `n` (μεγαλύτερο του 1) σαν παράμετρο. Διατυπώστε και μία συνάρτηση `ds3`, η οποία να έχει την ίδια λειτουργικότητα με τις `ds1` και `ds2`, αλλά να είναι σαφώς περισσότερο αποδοτική από αυτές.

```
int ds1(int n)
{  int i, s;
   s = 0;
   for (i = 1 ; i <= n ; i++)
      if (n % i == 0)
         s += i;
   return s; }
```

```
int ds2(int n)
{  int i, s;
   s = n+1;
   for (i = 2 ; i <= n/2 ; i++)
      if (n % i == 0)
         s += i;
   return s; }
```

2. (α΄) Ορίστε μία συνάρτηση C με πρωτότυπο `int listlength(Listptr)`, η οποία να επιστρέφει στο όνομά της το πλήθος των κόμβων μίας λίστας ακεραίων.

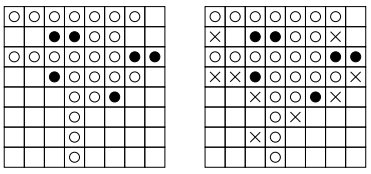
   (β΄) Ποια χρήσιμη λειτουργία κάνει η συνάρτηση C που δίνεται στη συνέχεια; Αν την καλούσατε για μία λίστα τριών ακεραίων, δείξτε σχηματικά ποια λίστα θα επιστρέψει η συνάρτηση αυτή στο όνομά της, χωρίς να παραθέσετε τα ενδιάμεσα βήματα κατά την κατασκευή της.

```
Listptr funlist(Listptr list)
{  Listptr tmplist, retlist;
   if (list == NULL || list->next == NULL)
      return list;
   retlist = tmplist = funlist(list->next);
   while (tmplist->next != NULL)
      tmplist = tmplist->next;
   list->next = NULL;
   tmplist->next = list;
   return retlist; }
```

3. Ένας θετικός ακέραιος λέγεται ότι είναι *ναρκισσιστικός αριθμός* (narcissistic number), όταν το πλήθος των ψηφίων του είναι k και ο αριθμός ισούται με το άθροισμα των ψηφίων του υψωμένων στη δύναμη k. Για παράδειγμα, ο αριθμός 153 είναι ναρκισσιστικός, αφού έχει 3 ψηφία και ισχύει $153 = 1^3 + 5^3 + 3^3$. Ομοίως και ο 1634 ($= 1^4 + 6^4 + 3^4 + 4^4$). Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`narcissistic.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (stdin) ένα θετικό ακέραιο N και να εκτυπώνει όλους τους ναρκισσιστικούς αριθμούς που είναι μικρότεροι ή ίσοι του N. Μία ενδεικτική εκτέλεση του προγράμματος είναι η εξής:

```
% ./narcissistic
Please, give maximum number: 1000000
1 2 3 4 5 6 7 8 9 153 370 371 407 1634 8208 9474 54748 92727 93084 548834
```

4. Το παιγνίδι *Othello* παίζεται σε ένα πλαίσιο $n \times n$ (άρτιο n και συνήθως $n = 8$), συνοδευόμενο από ένα σύνολο κυκλικών πλακιδίων, τα οποία είναι μαύρα στη μία όψη τους και λευκά στην άλλη. Οι παίκτες παίζουν εναλλάξ. Ο ένας (μαύρος παίκτης) τοποθετεί στο πλαίσιο πλακίδια με την μαύρη όψη τους προς τα επάνω και ο άλλος (λευκός παίκτης) με τη λευκή προς τα επάνω. Ένας παίκτης επιτρέπεται να τοποθετήσει πλακίδιο του χρώματός του σε κάποια θέση στο πλαίσιο, όταν από τη θέση αυτή και σε τουλάχιστον μία ευθεία γραμμή (οριζόντια, κατακόρυφα ή διαγώνια) υπάρχουν συνεχόμενα πλακίδια του αντίθετου χρώματος και αμέσως μετά πλακίδιο του χρώματος του παίκτη. Για παράδειγμα, αν μία κατάσταση του παιγνιδιού είναι αυτή που φαίνεται στο πρώτο σχήμα δεξιά και είναι η σειρά του μαύρου παίκτη να παίξει, τότε οι θέσεις στις οποίες έχει δικαίωμα ο παίκτης να βάλει μαύρο πλακίδιο είναι αυτές που έχουν σημειωθεί με $\times$ στο δεύτερο σχήμα.



Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`othello.c`"), το οποίο αρχικά να δέχεται από τη γραμμή εντολής έναν ακέραιο N (διάσταση ενός παιγνιδιού Othello) και ένα χαρακτήρα, είτε `'b'` είτε `'w'`, που σημαίνει ποιος παίκτης έχει σειρά να παίξει (μαύρος ή λευκός, αντίστοιχα) σε δεδομένη κατάσταση του παιγνιδιού. Στη συνέχεια, το πρόγραμμα να διαβάζει από την πρότυπη είσοδο (stdin) ένα διδιάστατο πίνακα χαρακτήρων $N \times N$, ο οποίος να αναπαριστά μία κατάσταση του παιγνιδιού, όπου το `'b'` σημαίνει μαύρο πλακίδιο, το `'w'` λευκό πλακίδιο και το `'.'` σημαίνει κενό τετραγωνίδιο. Το πρόγραμμα να βρίσκει τις νόμιμες κινήσεις του παίκτη που έχει σειρά να παίξει και να εκτυπώνει την κατάσταση του παιγνιδιού με επισημειωμένες αυτές τις θέσεις με ένα `'*'`. Ενδεικτικές εκτελέσεις του προγράμματος φαίνονται στη συνέχεια.

```
% ./othello 8 b
Please, give the Othello board
wwwwwww.
..bbww..
wwwwwwbb
..bwwww.
...wwb..
...w....
...w....
...w....
Legal moves of player b marked with *
wwwwwww.
*.bbww*.
wwwwwwbb
**bwwww*
..*wwb*.
...w*...
..*w....
...w....
```

```
% ./othello 6 w
Please, give the Othello board
..w.b.
..wb.b
.bbwwb
bbbwwb
..ww..
.www..
Legal moves of player w marked with *
..w*b.
**wb*b
*bbwwb
bbbwwb
.*ww..
.www..
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2017 (27/1/2017)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Η γραμμή ενός πηγαίου αρχείου C

   ```
   #include <stdlib.h>
   ```

   απευθύνεται στον ...
   ◯ Α. ... προεπεξεργαστή
   ◯ Β. ... μεταγλωττιστή
   ◯ Γ. ... συνδέτη

2. Η μνήμη που χρειάζεται για να αποθηκευθεί μία μεταβλητή τύπου `unsigned int` σε σχέση με τη μνήμη που χρειάζεται για να αποθηκευθεί μία μεταβλητή τύπου `signed int` είναι ...
   ◯ Α. ... περισσότερη
   ◯ Β. ... λιγότερη
   ◯ Γ. ... ακριβώς ίδια

3. Σε ποιο εύρος μπορούν να κυμανθούν οι τιμές μίας `signed char` μεταβλητής στη C;
   ◯ Α. `[-128,127]`
   ◯ Β. `[-127,128]`
   ◯ Γ. `[-128,128]`

4. Ο αριθμός `111` του δεκαδικού συστήματος αρίθμησης γράφεται στο δυαδικό σύστημα και στο δεκαεξαδικό σύστημα ως ...
   ◯ Α. ... `01111111` και `7F`, αντίστοιχα
   ◯ Β. ... `01101111` και `6F`, αντίστοιχα
   ◯ Γ. ... `11111111` και `FF`, αντίστοιχα

5. Αν η x είναι ακέραια μεταβλητή, ποια τιμή θα πάρει μετά την εντολή `x = (5 > 2) + (3 < 8) + (y = 4);`
   ◯ Α. `4`
   ◯ Β. `5`
   ◯ Γ. `6`

6. Για να πάρουμε το πηλίκο μίας `unsigned int` μεταβλητής με το 32, αρκεί να ...
   ◯ Α. ... ολισθήσουμε τη δυαδική αναπαράσταση της τιμής της πέντε θέσεις προς τα αριστερά, υπό την προϋπόθεση ότι τα bits που θα "χαθούν" θα είναι όλα μηδενικά
   ◯ Β. ... ολισθήσουμε τη δυαδική αναπαράσταση της τιμής της πέντε θέσεις προς τα δεξιά
   ◯ Γ. ... ολισθήσουμε τη δυαδική αναπαράσταση της τιμής της πέντε θέσεις προς τα δεξιά, υπό την προϋπόθεση ότι τα bits που θα "χαθούν" θα είναι όλα μηδενικά

7. Όταν μία εξωτερική μεταβλητή δηλωθεί σαν `static`, ...
   ◯ Α. ... φυλάσσεται στη στοίβα
   ◯ Β. ... φυλάσσεται στον σωρό
   ◯ Γ. ... η δήλωσή της ισχύει μόνο για το αρχείο μέσα στο οποίο ορίζεται

8. Τι είναι το `stdlib.h`;
   ◯ Α. Το αρχείο επικεφαλίδας της πρότυπης βιβλιοθήκης της C
   ◯ Β. Η πρότυπη βιβλιοθήκη της C
   ◯ Γ. Η πλήρης βιβλιοθήκη συναρτήσεων της C

9. Μία συμβολοσειρά στη C είναι ...
   ◯ Α. ... μία λίστα από χαρακτήρες
   ◯ Β. ... ένας πίνακας ακεραίων
   ◯ Γ. ... αδύνατον να τροποποιηθεί από κάποιο πρόγραμμα

10. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
signed char x = 120, y;
y = x + 10;
printf("%d\n", y);
```

◯ A. 130
◯ B. -126
◯ Γ. Τίποτα από τις άλλες επιλογές

11. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
#define diff(A,B) A-B
printf("%d\n", diff(8,2+1));
```

◯ A. 5
◯ B. 9
◯ Γ. 7

12. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = 7, y, *p;
p = &x;
y = *p++;
printf("%d %d\n", x, y);
```

◯ A. 7 7
◯ B. 7 8
◯ Γ. Τίποτα, γιατί θα προκληθεί σφάλμα κατά την εκτέλεση

13. Οι μακροεντολές στη C ορίζονται με ...
◯ A. ... #ifdef
◯ B. ... #define
◯ Γ. ... #ifndef

14. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = -1, y = 7, z = 5, w;
w = (x++) + (--y) * (++z);
if (x++) w++;
printf("%d\n", w);
```

◯ A. 30
◯ B. 35
◯ Γ. 36

15. Ποια τιμή επιστρέφει η κλήση συνάρτησης `strlen("0");`
◯ A. NULL
◯ B. 2
◯ Γ. 1

16. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int *p, x[] = {4, 3, 2, 1, 0};
p = x+3;
x[1] = *--p;
*p++ = --x[0];
*(p+1) = *p--;
p += 4;
p[-6]--;
p = x-1;
printf("%d %d %d %d %d %d\n", sizeof(x)/sizeof(int), x[0], x[1], x[2], x[3], x[4]);
```

◯ A. 1 3 2 3 1 0
◯ B. 5 2 2 3 1 1
◯ Γ. Τίποτα, γιατί θα προκληθεί σφάλμα κατά την εκτέλεση

17. Αν η συνάρτηση `f()` είναι ορισμένη ως

```
void f(int x)
{  if (x && (x-1) && (x-2)) {
       f(x-1);
       f(x-2);
       f(x-3); } }
```

και κάνουμε την κλήση `f(6)`, πόσες φορές θα κληθεί τελικά η `f(0)`;
◯ Α. 4
◯ Β. 5
◯ Γ. 6

18. Όταν η `main()` συνάρτηση ενός προγράμματος επιστρέψει την τιμή `0`, συμβατικά εννοούμε ότι ...
◯ Α. ... αυτό είναι το αποτέλεσμα του προγράμματος
◯ Β. ... το πρόγραμμα τερμάτισε επιτυχώς
◯ Γ. ... το πρόγραμμα τερμάτισε ανεπιτυχώς

19. Η συνάρτηση `strcmp()` της C ...
◯ Α. ... επιστρέφει σε περίπτωση λάθους το `-1`
◯ Β. ... επιστρέφει σε περίπτωση λάθους το `NULL`
◯ Γ. ... δεν υπάρχει περίπτωση να επιστρέψει με ένδειξη λάθους

20. Σε ποια εντολή από τις παρακάτω το `fun` είναι δείκτης σε συνάρτηση;
◯ Α. `int *fun(int *x);`
◯ Β. `int (*fun)(int *x);`
◯ Γ. `int **fun(int *x);`

21. Αν για κάποιο λόγο αποτύχει η κλήση της συνάρτησης `fopen()`, αυτή επιστρέφει ...
◯ Α. ... `EOF`
◯ Β. ... `NULL`
◯ Γ. ... `-1`

22. Αν ένα εκτελέσιμο πρόγραμμα, που έχει προκύψει από το πηγαίο πρόγραμμα C που ακολουθεί, κληθεί ως `./myprog 5 7` τι θα εκτυπώσει;

```
#include <stdio.h>
int main(int argc, char **argv)
{  printf("%s\n", *argv);
   return 0; }
```

◯ Α. `./myprog 5 7`
◯ Β. `myprog`
◯ Γ. `./myprog`

23. Σε μία απλά συνδεδεμένη λίστα, οι κόμβοι που ο δείκτης στον επόμενό τους ισούται με `NULL` είναι ...
◯ Α. ... τουλάχιστον ένας
◯ Β. ... ακριβώς ένας
◯ Γ. ... το πολύ ένας

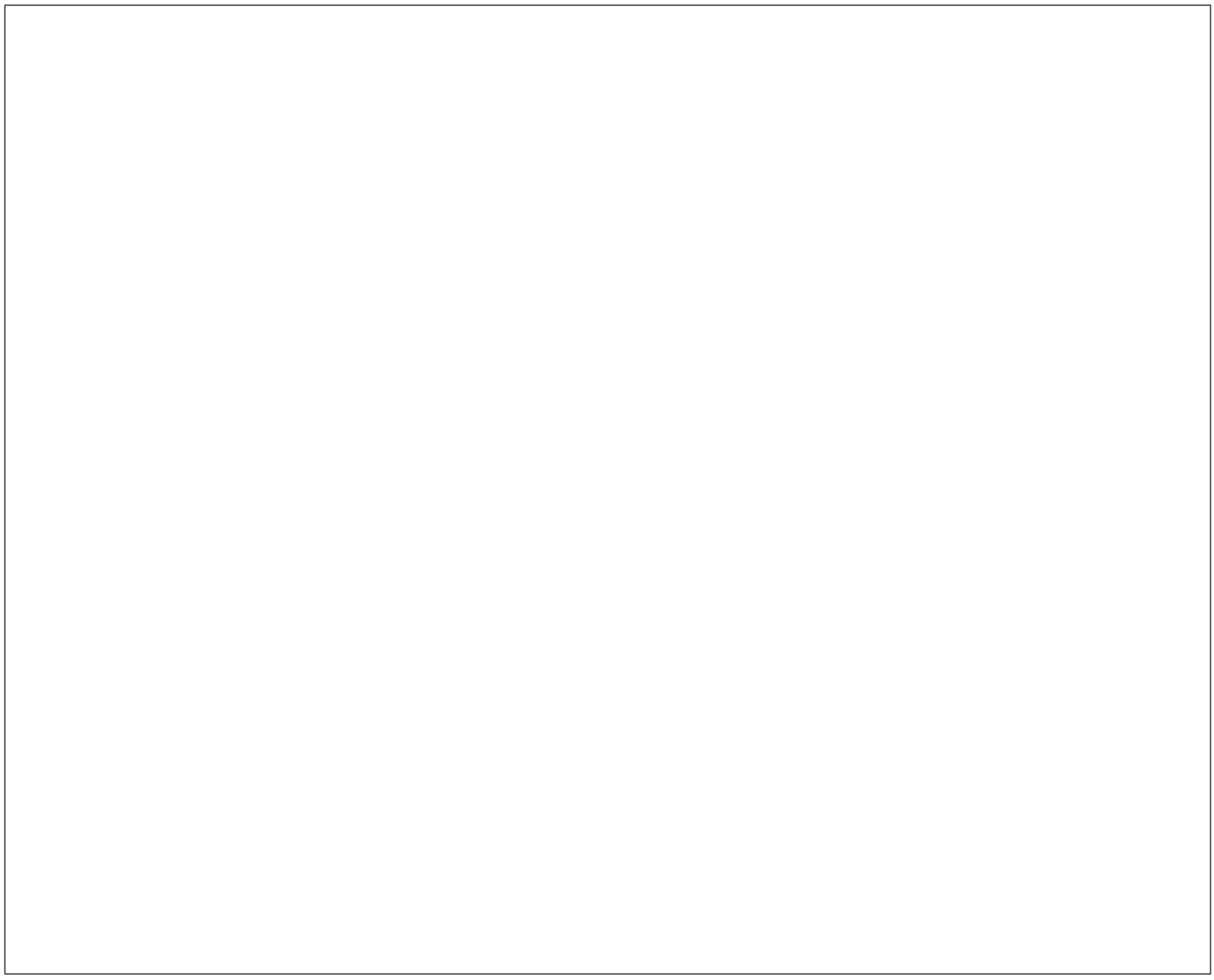
24. Σε ένα ταξινομημένο δυαδικό δέντρο, τα βάθη του αριστερού και του δεξιού υποδέντρων του κόμβου-ρίζας ...
◯ Α. ... είναι πάντοτε ίσα
◯ Β. ... είναι πάντοτε άνισα
◯ Γ. ... ενδεχομένως να είναι ίσα

25. Ο ταχύτερος αλγόριθμος για να ελέγξουμε αν ένας πίνακας είναι ταξινομημένος έχει πολυπλοκότητα ...
◯ Α. ... $O(n \cdot logn)$
◯ Β. ... $O(n)$
◯ Γ. ... $O(logn)$

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Γράψτε πρόγραμμα C το οποίο θα δέχεται στη γραμμή εντολής έναν ακέραιο N μεγαλύτερο του 1 και θα εκτυπώνει στην έξοδο, σε αύξουσα σειρά, όλους τους ακεραίους που είναι μικρότεροι ή ίσοι του N και οι οποίοι μπορούν να γραφούν σαν άθροισμα δύο κύβων με τουλάχιστον δύο διαφορετικούς τρόπους. Στην εκτύπωση να φαίνονται, για κάθε αριθμό, και τουλάχιστον δύο τρόποι που μπορεί να γραφεί αυτός σαν άθροισμα δύο κύβων. Για παράδειγμα, ο αριθμός 1729 είναι τέτοιος αριθμός, αφού $1729 = 1^3 + 12^3 = 9^3 + 10^3$. Σημειώστε ότι **δεν επιτρέπεται να χρησιμοποιήσετε πραγματικούς αριθμούς, συναρτήσεις της μαθηματικής βιβλιοθήκης και πίνακες** (εξαιρείται το όρισμα `argv[]` της `main()`). Μία ενδεικτική εκτέλεση:

```
$ ./taxicab 100000
 1729 =  1^3 + 12^3 =  9^3 + 10^3
 4104 =  2^3 + 16^3 =  9^3 + 15^3
13832 =  2^3 + 24^3 = 18^3 + 20^3
20683 = 10^3 + 27^3 = 19^3 + 24^3
32832 =  4^3 + 32^3 = 18^3 + 30^3
39312 =  2^3 + 34^3 = 15^3 + 33^3
40033 =  9^3 + 34^3 = 16^3 + 33^3
46683 =  3^3 + 36^3 = 27^3 + 30^3
64232 = 17^3 + 39^3 = 26^3 + 36^3
65728 = 12^3 + 40^3 = 31^3 + 33^3
```



Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Θεωρήστε ότι ένας κόμβος λίστας ακεραίων ορίζεται, κατά τα γνωστά, ως:

```
typedef struct listnode *Listptr;
struct listnode {
  int value;
  Listptr next; };
```

Η ελλιπής συνάρτηση που δίνεται στη συνέχεια δέχεται ως όρισμα έναν ακέραιο `n` και τη διεύθυνση μίας θέσης μνήμης `listaddr` στην οποία είναι καταχωρημένη η διεύθυνση του πρώτου κόμβου μίας λίστας ακεραίων. Η λειτουργία της συνάρτησης είναι να διαγράψει τον `n`-οστό κόμβο της λίστας, αποδεσμεύοντας και την αντίστοιχη μνήμη (η αρίθμηση των κόμβων ξεκινά από το 1). Αν η λίστα έχει λιγότερους από `n` κόμβους, ή το `n` δεν είναι θετικό, η συνάρτηση δεν διαγράφει κανένα. Συμπληρώστε τις θέσεις με τις τελείες (όχι κατ΄ ανάγκη όλες, αλλά, σίγουρα, όχι περισσότερες από όσες δίνονται) ώστε η συνάρτηση να λειτουργεί με τον τρόπο που περιγράφηκε. Δεν επιτρέπεται να χρησιμοποιήσετε άλλες μεταβλητές, πέραν αυτών που ορίζονται ήδη στη συνάρτηση.

```
void pdelnth(int n, Listptr *listaddr)
{  Listptr templist;
   while (.......................)
      if (..............) {
         ......................
         ......................
         ......................
         return; }
      else
         ..................... }
```

Επίσης, υλοποιήστε μία παραλλαγή της συνάρτησης αυτής (είτε επαναληπτικά, είτε αναδρομικά), με πρωτότυπο `Listptr delnth(int n, Listptr list)`, με την οποία να διαγράφεται ο `n`-οστός κόμβος της λίστας που η διεύθυνση του πρώτου κόμβου της είναι η τιμή της μεταβλητής `list`, αποδεσμεύοντας την αντίστοιχη μνήμη, και επιστρέφοντας τη νέα λίστα στο όνομα της συνάρτησης. Αν η λίστα έχει λιγότερους από `n` κόμβους, ή το `n` δεν είναι θετικό, η συνάρτηση δεν διαγράφει κανένα και επιστρέφει τη λίστα που της δόθηκε. Δεν επιτρέπεται να υλοποιήσετε την `delnth()` απλώς καλώντας κατάλληλα την `pdelnth()`.

Πώς θα πρέπει να κληθούν οι συναρτήσεις `pdelnth()` και `delnth()` από τη `main()` συνάρτηση ενός προγράμματος για να διαγράψουν, η πρώτη, το τρίτο στοιχείο και, η δεύτερη, το πέμπτο στοιχείο μίας λίστας ακεραίων, η διεύθυνση του πρώτου κόμβου της οποίας είναι η τιμή της μεταβλητής `alist`, τύπου `Listptr`;

Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Ορίστε δύο συναρτήσεις C με πρωτότυπα

```
unsigned int add(unsigned int a, unsigned int b)
```

και

```
unsigned int mult(unsigned int a, unsigned int b)
```

οι οποίες να επιστρέφουν το άθροισμα και το γινόμενο, αντίστοιχα, των ορισμάτων που τους δίνονται. Μία λεπτομέρεια είναι ότι δεν επιτρέπεται να χρησιμοποιήσετε στον κώδικα των συναρτήσεων τους χαρακτήρες `+`, `-`, `*`, `/`, `%`, `[` και `]`.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2018 (22/1/2018)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Ένα αντικειμενικό αρχείο περιέχει κώδικα ...
   ◯ Α. ... σε γλώσσα assembly
   ◯ Β. ... σε γλώσσα μηχανής που είναι άμεσα εκτελέσιμος
   ◯ Γ. ... σε γλώσσα μηχανής που δεν είναι άμεσα εκτελέσιμος

2. Υπό την προϋπόθεση ότι το πηγαίο αρχείο `aprog.c` δεν έχει συντακτικά λάθη, με την εντολή `gcc aprog.c` παράγεται ...
   ◯ Α. ... το αντικειμενικό αρχείο `aprog.o`
   ◯ Β. ... το εκτελέσιμο αρχείο `a.out`
   ◯ Γ. ... το εκτελέσιμο αρχείο `aprog`

3. Ποιος δεν είναι σωστός αριθμός στο οκταδικό σύστημα αρίθμησης;
   ◯ Α. Ο αριθμός 32087
   ◯ Β. Ο αριθμός 51110
   ◯ Γ. Ο αριθμός 77777

4. Για να έχει νόημα η εντολή `x = (3.14 > 2.71);`, πώς πρέπει να έχει δηλωθεί η μεταβλητή x;
   ◯ Α. `int x;`
   ◯ Β. `float x;`
   ◯ Γ. `double x;`

5. Η εντολή `x = 5++;` ...
   ◯ Α. ... είναι ισοδύναμη με την εντολή `x = 6;`
   ◯ Β. ... θα προκαλέσει λάθος κατά την εκτέλεσή της
   ◯ Γ. ... είναι συντακτικά λανθασμένη

6. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
signed char x = 1;
while (x > 0)
   x = (x << 1) | 1;
printf("%d\n", x);
```

   ◯ Α. 0
   ◯ Β. -1
   ◯ Γ. Τίποτα από τις άλλες επιλογές

7. Με τη συνάρτηση `malloc` ...
   ◯ Α. ... δεσμεύεται χώρος για τον κώδικα του προγράμματος
   ◯ Β. ... δεσμεύεται χώρος για δεδομένα στη στοίβα
   ◯ Γ. ... δεσμεύεται χώρος για δεδομένα στον σωρό

8. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = 5, y;
double z, w;
z = (double) x;
y = (int) z;
w = (double) (y / (2 * x));
printf("%.1f\n", w);
```

   ◯ Α. 0.5
   ◯ Β. 0.0
   ◯ Γ. 1.0

9. Μετά την εκτέλεση της εντολής `x = (132 == 0132);`, ποια είναι η τιμή της ακέραιας μεταβλητής x;
   ◯ Α. `132`
   ◯ Β. `1`
   ◯ Γ. `0`

10. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = 3, y = 5;
if (x == 3) if (y == 6) printf("one");
else printf("two");
printf("three\n");
```

   ◯ Α. `onetwothree`
   ◯ Β. `twothree`
   ◯ Γ. `three`

11. Η `main` συνάρτηση ενός προγράμματος ορίζεται ως `int main(void)` όταν ...
   ◯ Α. ... δεν μας ενδιαφέρει να εκτελέσουμε το πρόγραμμα με ορίσματα στη γραμμή εντολών
   ◯ Β. ... δεν επιστρέφει το πρόγραμμα κάποιο κωδικό εξόδου
   ◯ Γ. ... δεν χρειαζόμαστε χώρο στη στοίβα για την εκτέλεση του προγράμματος

12. Ποια είναι η σωστή δήλωση για τη μεταβλητή `fun`, ώστε να είναι συντακτικά σωστή η εντολή `fun = strcmp;`, όπου `strcmp` είναι η γνωστή συνάρτηση βιβλιοθήκης της C για τη σύγκριση δύο συμβολοσειρών;
   ◯ Α. `int fun(char *, char *);`
   ◯ Β. `int *fun(char *, char *);`
   ◯ Γ. `int (*fun)(char *, char *);`

13. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x, y = 5, z = 3, w;
x = y++ + ++z;
w = --x + y-- * ++z;
printf("%d\n", w);
```

   ◯ Α. `70`
   ◯ Β. `38`
   ◯ Γ. Τίποτα από τις άλλες επιλογές

14. Τι θα εκτυπωθεί από την παρακάτω εντολή;

```
printf("%c %c\n", 'e'-('u'-'U'), 'E'+('u'-'U'));
```

   ◯ Α. `E e`
   ◯ Β. `E E`
   ◯ Γ. `e e`

15. Αν η έξοδος του εκτελέσιμου προγράμματος που προκύπτει από το πηγαίο πρόγραμμα C που ακολουθεί (έστω με όνομα `myprog`) είναι η `1 2 3 4 5`, πώς θα έπρεπε να είχε εκτελεσθεί αυτό;

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{  int *x;
   if (argc == 1) return 1;
   if ((x = malloc((argc - 1) * sizeof(int))) == NULL) return 2;
   while (--argc)
      x[argc-1] = atoi(argv[argc]);
   while (argv[++argc] != NULL)
      printf("%d ", x[argc-1]);
   printf("\n");
   return 0; }
```

   ◯ Α. `./myprog 5 4 3 2 1`
   ◯ Β. `./myprog 2 3 4 5`
   ◯ Γ. `./myprog 1 2 3 4 5`

16. Μπορούμε σε ένα πρόγραμμα C, που χρησιμοποιεί ένα πίνακα 10 ακεραίων με όνομα `x`, να αντικαταστήσουμε τη δήλωση `int x[10]` με την `int *x`, χωρίς να αλλάξουμε κάτι άλλο στον υπόλοιπο κώδικα. Συμφωνείτε;
    - ◯ Α. Ναι, αρκεί να εκτελέσουμε πρώτα την εντολή `x++`
    - ◯ Β. Ναι, χωρίς κανένα πρόβλημα
    - ◯ Γ. Όχι, γιατί θα έχουμε συντακτικά λάθη ή/και λάθη κατά την εκτέλεση ή/και λάθος αποτελέσματα

17. Η παράσταση `*p->next++` **δεν** είναι ισοδύναμη με την ...
    - ◯ Α. ... `(*p->next)++`
    - ◯ Β. ... `*(p->next++)`
    - ◯ Γ. ... `*(p->next)++`

18. Η συνάρτηση `printf` της C μπορεί να υλοποιηθεί μέσω της ...
    - ◯ Α. ... `sprintf`
    - ◯ Β. ... `fprintf`
    - ◯ Γ. ... `scanf`

19. Για να έχουμε πρόσβαση στους κόμβους ενός ταξινομημένου δυαδικού δέντρου ...
    - ◯ Α. ... αρκεί να γνωρίζουμε τη διεύθυνση της ρίζας του
    - ◯ Β. ... αρκεί να γνωρίζουμε τη διεύθυνση του ελάχιστου στοιχείου του
    - ◯ Γ. ... αρκεί να γνωρίζουμε τη διεύθυνση του μέγιστου στοιχείου του

20. Τι θα εκτυπωθεί από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
void g(int);
void f(int x)
{  if (x < 2) return;
   g(x-1);
   g(x-2); }
void g(int x)
{  if (x > 1)
      f(x-1);
   printf("*"); }
int main(void)
{  f(5); printf("\n"); return 0; }
```

    - ◯ Α. `****`
    - ◯ Β. `*****`
    - ◯ Γ. `******`

21. Τις απαριθμήσεις στη C τις διαχειρίζεται ο ...
    - ◯ Α. ... προεπεξεργαστής
    - ◯ Β. ... μεταγλωττιστής
    - ◯ Γ. ... συνδέτης

22. Τις μακροεντολές στη C τις διαχειρίζεται ο ...
    - ◯ Α. ... προεπεξεργαστής
    - ◯ Β. ... μεταγλωττιστής
    - ◯ Γ. ... συνδέτης

23. Αν για ένα πρόβλημα υπάρχουν δύο αλγόριθμοι Α και Β, όπου οι πολυπλοκότητες χρόνου τους είναι $O(n^2 \cdot logn)$ και $O(n \cdot (logn)^2)$, αντίστοιχα, τότε ...
    - ◯ Α. ... ο αλγόριθμος Α είναι καλύτερος από τον Β
    - ◯ Β. ... ο αλγόριθμος Β είναι καλύτερος από τον Α
    - ◯ Γ. ... οι δύο αλγόριθμοι είναι εξ ίσου καλοί

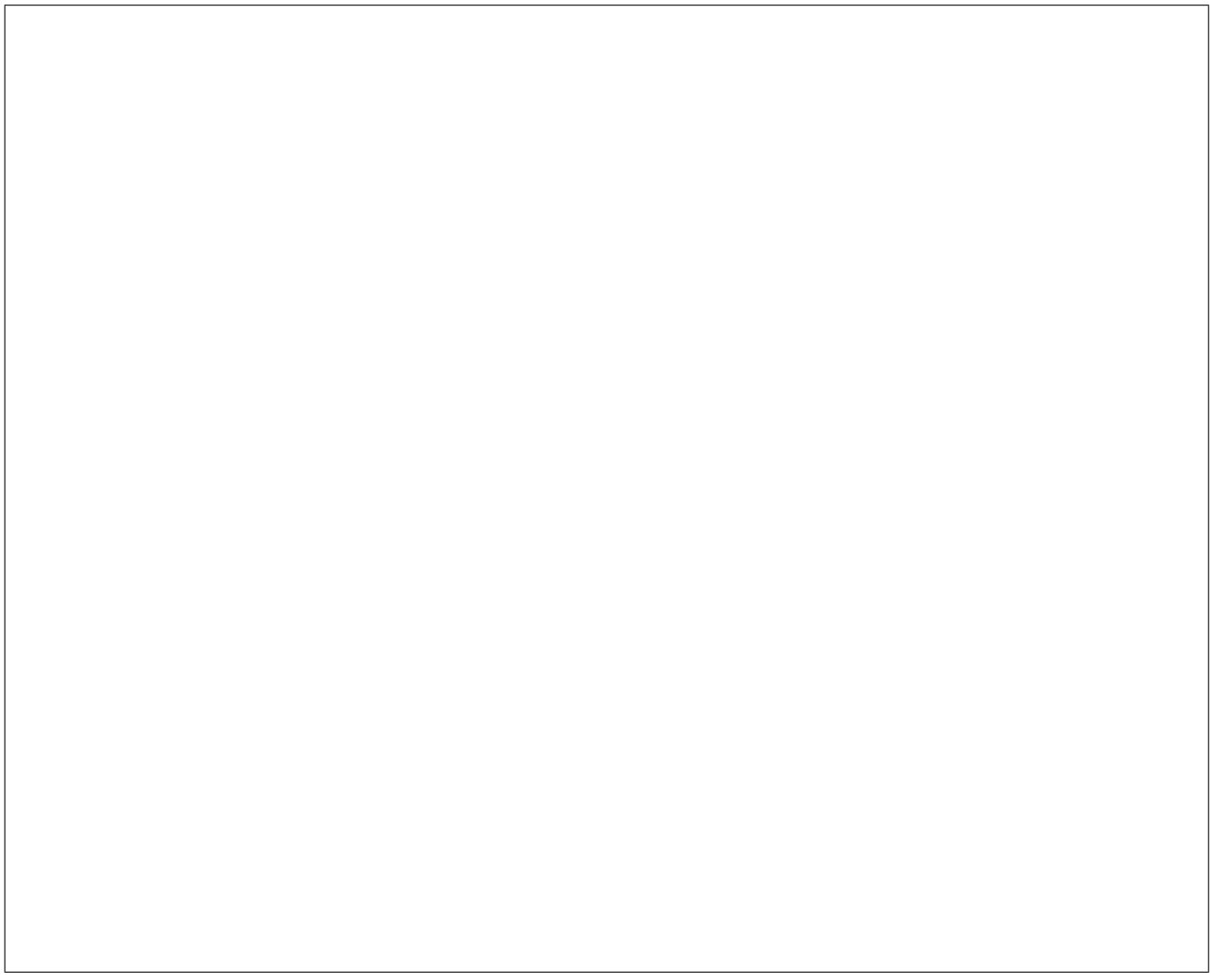
24. Ποια είναι η πολυπλοκότητα της μεθόδου ταξινόμησης της επιλογής;
    - ◯ Α. $O(n \cdot logn)$
    - ◯ Β. $O(n^2)$
    - ◯ Γ. $O(n^3)$

25. Το πρόγραμμα για την έκδοση των αποτελεσμάτων των πανελληνίων εξετάσεων μπορεί να υλοποιηθεί σε C. Συμφωνείτε;
    - ◯ Α. Ναι, φυσικά, χωρίς κανένα πρόβλημα
    - ◯ Β. Όχι, γιατί χρειάζονται εξειδικευμένες γλώσσες για τέτοιου είδους εφαρμογές
    - ◯ Γ. Ναι, αρκεί να το αναθέσουμε σε κάποιον υψηλού επιπέδου προγραμματιστή που εργάζεται στην Google

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Ένας θετικός ακέραιος αριθμός n ονομάζεται k-*υπερτέλειος* (k-hyperperfect) όταν υπάρχει θετικός ακέραιος k τέτοιος ώστε να ισχύει $n = 1+k\cdot(\sigma(n)-n-1)$, όπου $\sigma(n)$ είναι το άθροισμα των διαιρετών του n. Στην ειδική περίπτωση που έχουμε $k = 1$, ο αριθμός n ονομάζεται *τέλειος* (perfect). Για παράδειγμα, ο αριθμός 21 είναι 2-υπερτέλειος, αφού $\sigma(21) = 1+3+7+21 = 32$ και $21 = 1+2\cdot(32-21-1)$, ενώ ο αριθμός 28 είναι τέλειος, αφού $\sigma(28) = 1+2+4+7+14+28 = 56$ και $28 = 1+1\cdot(56-28-1)$. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`khperf.c`"), το οποίο να παίρνει από τη γραμμή εντολής δύο θετικούς ακεραίους M και N ($M \leq N$) και να βρίσκει όλους τους k-υπερτέλειους αριθμούς στο διάστημα $[M, N]$ που **δεν** είναι τέλειοι. Σημειώστε ότι **δεν επιτρέπεται να χρησιμοποιήσετε πραγματικούς αριθμούς, συναρτήσεις της μαθηματικής βιβλιοθήκης και πίνακες** (εξαιρείται το όρισμα `argv[]` της `main()`). Επίσης, θα εκτιμηθούν περισσότερο απαντήσεις με καλή χρονική απόδοση. Μία ενδεικτική εκτέλεση:

```
$ ./khperf 60000000 70000000
60110701 is 6-hyperperfect
61442077 is 3918-hyperperfect
61599553 is 672-hyperperfect
62722153 is 12-hyperperfect
62804941 is 2838-hyperperfect
63519481 is 3852-hyperperfect
64220161 is 1536-hyperperfect
65618101 is 4050-hyperperfect
```



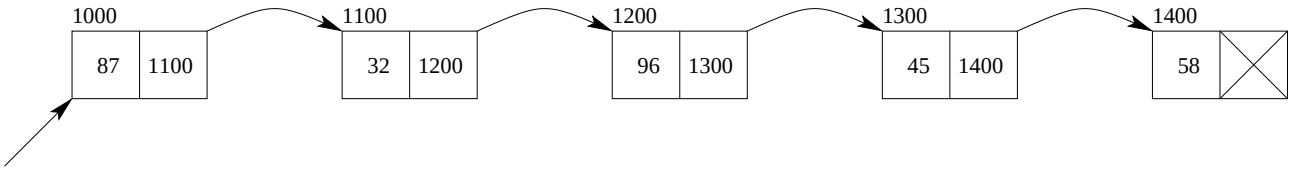
Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Δίνεται η παρακάτω συνάρτηση C, καθώς και ο συνήθης ορισμός ενός κόμβου λίστας ακεραίων:

```
Listptr sl(Listptr list)
{  Listptr el, fl, tl;
   if (list == NULL) return NULL;
   el = list->next;
   fl = list->next;
   while (fl != NULL) {
      tl = list->next;
      list->next = fl->next;
      fl = fl->next;
      list = tl; }
   return el; }
```

```
typedef struct listnode *Listptr;

struct listnode {
  int value;
  Listptr next;
};
```

Αν η συνάρτηση `sl()` κληθεί με όρισμα τη διεύθυνση του πρώτου κόμβου της λίστας που φαίνεται στη συνέχεια, δηλαδή το 1000, τι θα επιστρέψει στο όνομά της; Δείξτε διαδοχικά τις τιμές που θα παίρνουν οι μεταβλητές της συνάρτησης και σχεδιάστε την τελική κατάσταση της μνήμης μετά την κλήση της συνάρτησης. Περιγράψτε **σε δύο γραμμές το πολύ** τη λειτουργικότητα της συνάρτησης.



Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Τι πιστεύετε ότι θα εκτυπωθεί από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{  int i = 0, n;
   double x, y;
   for (n = 0 ; n < 1000000 ; n++)  {
      x = 2.0 * rand() / RAND_MAX - 1;
      y = 2.0 * rand() / RAND_MAX - 1;
      if (x * x + y * y < 1)
         i++;
   }
   printf("%.2f\n", 4.0 * i / n);
   return 0;
}
```

Αιτιολογήστε την απάντησή σας.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2019 (21/1/2019)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Για τη μετατροπή ενός πηγαίου αρχείου σε εκτελέσιμο, ...
   - ◯ Α. ... χρειάζεται ο μεταγλωττιστής
   - ◯ Β. ... χρειάζεται ο συνδέτης
   - ◯ Γ. ... χρειάζονται τόσο ο μεταγλωττιστής, όσο και ο συνδέτης
2. Αν το αρχείο `myprog.exe` περιέχει εκτελέσιμο κώδικα και έχει δημιουργηθεί σε περιβάλλον Windows, ...
   - ◯ Α. ... τότε μπορεί να εκτελεσθεί σε περιβάλλον Linux, αν το μετονομάσουμε σε `myprog` και το καλέσουμε ως `./myprog` στη γραμμή εντολών
   - ◯ Β. ... τότε δεν μπορεί να εκτελεσθεί σε περιβάλλον Linux, απλώς καλώντας το στη γραμμή εντολών, όποιο όνομα και να έχει
   - ◯ Γ. ... τότε μπορεί να εκτελεσθεί σε περιβάλλον Linux, αν μετατραπεί σε κατάλληλη μορφή από τον συνδέτη του Linux
3. Αν τα πλήθη των διαφορετικών ακεραίων που μπορούν να φυλαχθούν σε μία `signed int` μεταβλητή και μία `unsigned int` μεταβλητή είναι S και U αντίστοιχα, ποιο από τα παρακάτω ισχύει;
   - ◯ Α. $S < U$
   - ◯ Β. $S = U$
   - ◯ Γ. $S > U$
4. Αν το `i` είναι ακέραια μεταβλητή, τότε οι παραστάσεις `1/(((double) i) * ((double) i))` και `1/((double) (i * i))` παίρνουν την ίδια τιμή, έστω και κατά προσέγγιση.
   - ◯ Α. Μερικές φορές ναι, άλλες φορές όχι
   - ◯ Β. Ναι, ισχύει πάντα
   - ◯ Γ. Ποτέ δεν ισχύει
5. Σε μία μεταβλητή τύπου `double`, μπορεί να φυλαχθεί οποιοσδήποτε πραγματικός αριθμός με απόλυτη ακρίβεια, υπό την προϋπόθεση ότι βρίσκεται μέσα στο διάστημα που καθορίζεται από την ελάχιστη και τη μέγιστη τιμή που μπορεί να φυλαχθούν στη μεταβλητή.
   - ◯ Α. Σωστά, εφόσον ο αριθμός δεν είναι άρρητος
   - ◯ Β. Σωστά, εφόσον ο αριθμός είναι θετικός
   - ◯ Γ. Λάθος
6. Αν η ακέραια μεταβλητή `x` έχει την τιμή 2, ποια τιμή θα πάρει η ακέραια μεταβλητή `y` μετά την εκτέλεση της εντολής `y = (x+5)++;`
   - ◯ Α. 7
   - ◯ Β. 8
   - ◯ Γ. Η εντολή είναι συντακτικά λανθασμένη
7. Τι θα εκτυπωθεί από την εντολή `printf("%d", 73 | (100 & 52));`
   - ◯ Α. 1
   - ◯ Β. 109
   - ◯ Γ. Η εντολή είναι συντακτικά λανθασμένη
8. Πού φυλάσσονται οι εξωτερικές (ή καθολικές) μεταβλητές;
   - ◯ Α. Στον χώρο των στατικών δεδομένων
   - ◯ Β. Στη στοίβα
   - ◯ Γ. Στον σωρό
9. Αν έχουμε δεσμεύσει με τη συνάρτηση `malloc` χώρο για ένα πίνακα και στη μεταβλητή `p` έχουμε κρατήσει το δείκτη στο πρώτο στοιχείο του πίνακα, τότε η εντολή `free(++p); ...`
   - ◯ Α. ... είναι συντακτικά λανθασμένη
   - ◯ Β. ... πιθανότατα θα προκαλέσει λάθος κατά την εκτέλεση
   - ◯ Γ. ... θα αποδεσμεύσει τη μνήμη για όλα τα στοιχεία του πίνακα πλην του πρώτου

10. Ποια τιμή θα επιστρέψει η κλήση της συνάρτησης `strlen("Hello world!");`
    ◯ Α. 11
    ◯ Β. 12
    ◯ Γ. 13

11. Αν η κωδικοποίηση των χαρακτήρων είναι σύμφωνα με το ASCII πρότυπο, πόσα bytes απαιτούνται για να φυλαχθεί η συμβολοσειρά `"Hello world!"` στη μνήμη;
    ◯ Α. 11
    ◯ Β. 12
    ◯ Γ. 13

12. Η εντολή `while (i < 100)` είναι ισοδύναμη με την εντολή ...
    ◯ Α. `... for ( ; i < 100 ; )`
    ◯ Β. `... for (i = 0; i < 100 ; i++)`
    ◯ Γ. `... for ( ; i < 100 ; i++)`

13. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = 7, y = 1, z = 0;
while (x-- != ++y)
   z++;
printf("%d\n", z);
```

    ◯ Α. 3
    ◯ Β. 4
    ◯ Γ. Τίποτα, γιατί ο κώδικας θα εκτελείται επ΄ άπειρον

14. Αν το εκτελέσιμο πρόγραμμα που προκύπτει από το πηγαίο πρόγραμμα C που ακολουθεί (έστω με όνομα `myprog`) κληθεί ως `./myprog ab cde fghij`, τι θα εκτυπωθεί στην έξοδο;

```
#include <stdio.h>
int main(int argc, char **argv) {
   int z = 0;
   while (argv[z++] != NULL);
   printf("%d\n", z);
   return 0; }
```

    ◯ Α. 3
    ◯ Β. 4
    ◯ Γ. 5

15. Ποια θα είναι τα περιεχόμενα του πίνακα `x` μετά την εκτέλεση του παρακάτω τμήματος κώδικα C;

```
int *p, x[] = {5, 7, 2, 3, 6, 0, 1, 4};
p = x;
do {
   ++p;
} while (*(p-1) = *p);
```

    ◯ Α. 7 2 3 6 0 0 0 0
    ◯ Β. 7 2 3 6 0 0 1 4
    ◯ Γ. 5 7 2 3 6 0 0 1

16. Αν η `x` είναι μεταβλητή τύπου `unsigned int`, τότε η εντολή `x >>= 3;` είναι ισοδύναμη με την εντολή ...
    ◯ Α. `... x = x / 3;`
    ◯ Β. `... x /= 8;`
    ◯ Γ. `... x = 8 * x`

17. Αν επιθυμούμε το εκτελέσιμο που θα προκύψει από ένα πηγαίο πρόγραμμα C να επεξεργασθεί τα ορίσματα που θα δεχθεί από τη γραμμή εντολών, τότε η `main()` συνάρτηση του προγράμματος θα μπορούσε να ορισθεί ως ...
    ◯ Α. `... int main(int x, char *y[])`
    ◯ Β. `... int main(int argc, char argv[])`
    ◯ Γ. `... int main(char **argv, int argc)`

18. Ζητήθηκε από κάποιον να γράψει ένα πρόγραμμα C το οποίο θα υπολογίζει το άθροισμα (`sum`) των διαιρετών ενός ακεραίου αριθμού (`number`) και το σχετικό τμήμα του κώδικά του ήταν το εξής:

```
for (sum = 0, divisor = 2 ; divisor * divisor < number ; divisor++)
   if (!(number % divisor))
      sum += divisor + number / divisor;
```

◯ A. Ο κώδικας είναι σωστός
◯ B. Ο κώδικας δεν είναι σωστός, αλλά μπορεί να διορθωθεί αν στη συνθήκη τερματισμού του `for` αντικατασταθεί ο τελεστής `<` με τον `<=`
◯ Γ. Ο κώδικας δεν είναι σωστός, αλλά μπορεί να διορθωθεί αν μετά την εντολή `for` προστεθεί και η εντολή `if (divisor * divisor == number) sum += divisor;`

19. Τι θα εκτυπωθεί από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
int count;
void f(int n) {
   if (n < 3) return;
   count++;
   f(n-1); f(n-2); f(n-3); }
int main() {
   f(6);
   printf("%d \n", count);
   return 0; }
```

◯ A. 8
◯ B. 25
◯ Γ. Τίποτα, γιατί το πρόγραμμα θα εκτελείται επ΄ άπειρον

20. Ποιος είναι ο μέγιστος αριθμός κόμβων που μπορεί να υπάρχουν σε ένα δυαδικό δέντρο με d επίπεδα;
◯ A. 2^d
◯ B. $2^d - 1$
◯ Γ. 2^{d-1}

21. Ποια είναι η πολυπλοκότητα χρόνου για την εισαγωγή ενός κόμβου στο τέλος μίας απλά συνδεδεμένης λίστας με n κόμβους;
◯ A. $O(n)$
◯ B. $O(n^2)$
◯ Γ. $O(logn)$

22. Το μέγιστο στοιχείο ενός ταξινομημένου δυαδικού δέντρου ...
◯ A. ... είναι, σε κάθε περίπτωση, η ρίζα του
◯ B. ... βρίσκεται, σε κάθε περίπτωση, στο τελευταίο επίπεδο του δέντρου
◯ Γ. ... είναι πιθανό να έχει κόμβους απογόνους

23. Η μέθοδος ταξινόμησης του σωρού ...
◯ A. ... είναι επαναληπτική
◯ B. ... είναι αναδρομική
◯ Γ. ... απαιτεί δυναμική δέσμευση μνήμης για να λειτουργήσει σωστά

24. Αν αναζητήσετε στο pdf αρχείο των σημειώσεων/διαφανειών του μαθήματος τη λέξη `goto`, πόσες φορές πιστεύετε ότι θα την βρείτε;
◯ A. Καμία
◯ B. 5
◯ Γ. 50

25. Το τμήμα που σπουδάζετε γίνεται φέτος ...
◯ A. ... 20 ετών
◯ B. ... 30 ετών
◯ Γ. ... 40 ετών

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Μπορεί να αποδειχθεί μαθηματικά ότι ισχύει η σχέση

$$\frac{\sqrt{2}}{2}\cdot\frac{\sqrt{2+\sqrt{2}}}{2}\cdot\frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}\cdot\frac{\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}{2}\cdot\ldots\ldots\ldots\ldots=\frac{2}{\pi}$$

όπου π είναι το γνωστό $3.14\ldots$ Υλοποιήστε σε C ένα πρόγραμμα που θα υπολογίζει με τη βοήθεια της προηγούμενης σχέσης το π με ακρίβεια 10^{-7}. Μπορείτε να βρείτε το π υπολογίζοντας το γινόμενο πεπερασμένου πλήθους (έστω k) κλασμάτων της σχέσης, μέχρις ότου η διαφορά δύο διαδοχικών υπολογισμών γινομένων (με $k-1$ και k κλάσματα) γίνει μικρότερη από την επιθυμητή ακρίβεια. Το πρόγραμμά σας να εκτυπώνει εκτός από την προσεγγιστική τιμή που βρήκε για το π και το πλήθος των κλασμάτων που λήφθηκαν υπόψη για τον υπολογισμό (δηλαδή το k). Σημειώστε ότι **δεν επιτρέπεται να χρησιμοποιήσετε πίνακες ή δείκτες**. Μία ενδεικτική εκτέλεση:

```
$ ./pi
Computed pi = 3.14159258 by multiplying 12 fractions
```

Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Για να χρησιμοποιήσουμε στη C μία λίστα ακεραίων, συνήθως ορίζουμε τον κόμβο της ως εξής:

```
typedef struct listnode *Listptr;
struct listnode {
   int value;
   Listptr next; }
```

i. Γράψτε δύο εκδοχές (μία αναδρομική και μία επαναληπτική) μίας συνάρτησης C, η οποία θα παίρνει σαν παράμετρο μία λίστα ακεραίων και θα απελευθερώνει (μέσω της συνάρτησης `free`) τη μνήμη που είχε δεσμευθεί για τους κόμβους της λίστας.

ii. Γράψτε μία συνάρτηση C που να διαβάζει από την πρότυπη είσοδο (stdin) ακεραίους, μέχρι το τέλος της εισόδου (`EOF`), και να επιστρέφει με κάποιο τρόπο μία λίστα με αυτούς τους ακεραίους, με αντίστροφη σειρά από αυτήν που εισήχθησαν. Δεν επιτρέπεται να χρησιμοποιήσετε κάποια από τις γνωστές σας συναρτήσεις διαχείρισης λιστών που υπάρχουν στις σημειώσεις/διαφάνειες του μαθήματος. Επίσης, υποθέστε ότι οποιαδήποτε απόπειρα δυναμικής δέσμευσης μνήμης από τη συνάρτησή σας θα είναι επιτυχημένη.

iii. Δώστε **μόνο** το πρωτότυπο μίας συνάρτησης C που θα προσαρτά μία λίστα στο τέλος μίας άλλης.

Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Σας δίνεται ο παρακάτω κώδικας C:

```
void g(int s, int n, int k, int m, int *c)
{  int i;
   if (k == m) {
      for (i = 0 ; i < m ; i++)
         printf("%d ", c[i] + 1);
      printf("\n"); }
   else {
      for (i = s ; i < n ; i++) {
         c[k] = i;
         g(i + 1, n, k + 1, m, c); } } }

void f(int n, int m)
{  int i, *c;
   c = malloc(m * sizeof(int));
   g(0, n, 0, m, c);
   free(c); }
```

i. Τι θα εκτυπωθεί στην έξοδο αν η συνάρτηση `f()` κληθεί ως `f(5,3)`;

ii. Πόσες γραμμές θα εκτυπωθούν στην έξοδο αν κληθεί ως `f(20,6)`;

Αιτιολογήστε στοιχειωδώς τις απαντήσεις σας. Σε κάθε περίπτωση, θεωρήστε ότι οι συναρτήσεις `malloc()` και `free()` που καλούνται επιστρέφουν με επιτυχία.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2020 (27/1/2020)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Το αποτέλεσμα της εντολής μεταγλώττισης `gcc -c myprog.c` είναι ένα αρχείο το οποίο ...
   ◯ Α. ... είναι άμεσα εκτελέσιμο
   ◯ Β. ... μπορεί να μετατραπεί σε εκτελέσιμο από τον συνδέτη
   ◯ Γ. ... μπορεί να μετατραπεί σε εκτελέσιμο από τον προεπεξεργαστή

2. Συμφωνείτε ότι αν σε ένα πρόγραμμα C υπάρχει η γραμμή `#define PI 3.14159`, τότε πρέπει αμέσως μετά να ακολουθεί και η δήλωση `double PI;`
   ◯ Α. Ναι, γιατί διαφορετικά θα προκύψει λάθος κατά τη μεταγλώττιση
   ◯ Β. Όχι, δεν πρέπει να ακολουθήσει η δήλωση, γιατί θα προκύψει λάθος κατά τη μεταγλώττιση
   ◯ Γ. Δεν υπάρχει κάποια επίπτωση, είτε προστεθεί είτε όχι η δήλωση

3. Αν το πλήθος των ακεραίων με πρόσημο που μπορούν να φυλαχθούν σε 2 bytes ισούται με K και το πλήθος των ακεραίων χωρίς πρόσημο που μπορούν να φυλαχθούν σε 4 bytes ισούται με M, τότε ισχύει ...
   ◯ Α. ... $M = K^2$
   ◯ Β. ... $M = 2 \cdot K$
   ◯ Γ. ... $M = 2^K$

4. Στο αρχείο επικεφαλίδας `math.h` περιέχεται, μεταξύ άλλων, και ...
   ◯ Α. ... η υλοποίηση της συνάρτησης `sqrt` σε πηγαία μορφή
   ◯ Β. ... η υλοποίηση της συνάρτησης `sqrt` σε γλώσσα μηχανής
   ◯ Γ. ... το πρωτότυπο `double sqrt(double);` της συνάρτησης `sqrt`

5. Ποια τιμή θα έχει η ακέραια μεταβλητή `x` μετά την εκτέλεση της εντολής `x = 14/5.0 + 14.0/5 + 14/5;`
   ◯ Α. 6
   ◯ Β. 7
   ◯ Γ. 8

6. Ποια τιμή θα έχει η ακέραια μεταβλητή `x` μετά την εκτέλεση της εντολής `x = 5^3;`
   ◯ Α. 6
   ◯ Β. 0
   ◯ Γ. 125

7. Συμφωνείτε ότι μία εξωτερική μεταβλητή απαιτεί περισσότερη μνήμη για την αποθήκευσή της σε σχέση με μία τοπική μεταβλητή ίδιου τύπου;
   ◯ Α. Ναι
   ◯ Β. Όχι, εξωτερικές και τοπικές μεταβλητές ίδιου τύπου απαιτούν την ίδια μνήμη για την αποθήκευσή τους
   ◯ Γ. Ισχύει, αλλά μόνο για `double` και `float` μεταβλητές

8. Ποια από τις παρακάτω εντολές θα εκτελείται επ΄ άπειρον;
   ◯ Α. `for (char ch = 'Z' ; ch > 0 ; ch--);`
   ◯ Β. `for (signed int i = 0 ; i >= 0 ; i++);`
   ◯ Γ. `for (unsigned int i = 0 ; i >= 0 ; i++);`

9. Μία ακέραια μεταβλητή μπορεί να λειτουργήσει και ως αληθής συνθήκη σε μία εντολή `if` αν η τιμή της είναι ...
   ◯ Α. ... μεγαλύτερη από το 0
   ◯ Β. ... ίση με το 0
   ◯ Γ. ... διαφορετική από το 0

10. Ποιο από τα παρακάτω ισχύει για την εντολή `switch`;
    ◯ Α. Δεν είναι απαραίτητο να υπάρχει η εντολή `break` μετά το τέλος των εντολών κάθε `case`
    ◯ Β. Η σειρά των ακεραίων που αντιστοιχούν σε κάθε `case` πρέπει να είναι αύξουσα
    ◯ Γ. Πρέπει να καλύπτεται και η περίπτωση `default` για το ενδεχόμενο να μην είναι εφαρμόσιμη καμία `case`

11. Αν έχουμε δεσμεύσει δυναμικά ένα μονοδιάστατο πίνακα ακεραίων, με τη διεύθυνση του πρώτου του στοιχείου να έχει αποθηκευτεί στη μεταβλητή `p` (δηλωμένη ως `int *p`) και θέλουμε να περάσουμε αυτόν τον πίνακα σε μία συνάρτηση με πρωτότυπο `void f(int *p)`, πώς πρέπει να καλέσουμε τη συνάρτηση;
   ◯ Α. `f(*p)`
   ◯ Β. `f(&p)`
   ◯ Γ. `f(p)`

12. Αν για τη φύλαξη μίας μεταβλητής `d`, που έχει δηλωθεί ως `double *d` απαιτούνται D bytes και για τη φύλαξη μίας μεταβλητής `f`, που έχει δηλωθεί ως `float **f` απαιτούνται F bytes, τότε ποιο από τα παρακάτω ισχύει;
   ◯ Α. $D = F$
   ◯ Β. $D = 2 \cdot F$
   ◯ Γ. $F = 2 \cdot D$

13. Ποια θα είναι τα περιεχόμενα του πίνακα `x` μετά την εκτέλεση του παρακάτω τμήματος κώδικα C;

```
int i, *p, x[] = {7, 2, 4, 8, 1, 3, 6, 5, 9};
p = x;
for (i = 0 ; i < sizeof(x)/sizeof(int) - 1 ; i += 2)
   x[i+1] = *p++;
```

   ◯ Α. `7 4 2 7 1 7 6 4 9`
   ◯ Β. `7 7 4 7 1 4 6 7 9`
   ◯ Γ. Καμία από τις προηγούμενες απαντήσεις δεν είναι σωστή

14. Αν το εκτελέσιμο πρόγραμμα που προκύπτει από το πηγαίο πρόγραμμα C που ακολουθεί (έστω με όνομα `myprog`) κληθεί ως `./myprog 5 8 2 4`, τι θα εκτυπωθεί στην έξοδο;

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
   int x = 0;
   while (--argc > 1)
      x += atoi(argv[argc]) * atoi(argv[argc-1]);
   printf("%d\n", x);
   return 0; }
```

   ◯ Α. `320`
   ◯ Β. `73`
   ◯ Γ. `64`

15. Πώς θα πρέπει να έχει δηλωθεί μία μεταβλητή `fun` ώστε να έχει νόημα να της αναθέσουμε ως τιμή τη συνάρτηση `sqrt` της μαθηματικής βιβλιοθήκης μέσω της εντολής `fun = sqrt;`
   ◯ Α. `(double *) fun(double);`
   ◯ Β. `double (*fun)(double);`
   ◯ Γ. `double *fun(double);`

16. Αν οι `x` και `y` είναι ακέραιες μεταβλητές, τότε η παράσταση `x >> y` ερμηνεύεται ως αληθής συνθήκη, όταν ...
   ◯ Α. ... η `x` είναι μεγαλύτερη από την `y`
   ◯ Β. ... η `x` είναι πολύ μεγαλύτερη από την `y`
   ◯ Γ. ... ολισθαίνοντας την `x` προς τα δεξιά κατά `y` bits, το αποτέλεσμα είναι μη μηδενικό

17. Αν μία τοπική μεταβλητή συνάρτησης δηλωθεί ως `static`, αυτό σημαίνει ότι η μεταβλητή αυτή ...
   ◯ Α. ... δεν θα φυλαχθεί στη στοίβα
   ◯ Β. ... θα φυλαχθεί στον σωρό
   ◯ Γ. ... δεν επιτρέπεται να μεταβληθεί από τη συνάρτηση

18. Αν η παρακάτω συνάρτηση κληθεί ως `f(4)`, τι θα εκτυπωθεί στην έξοδο;

```
void f(int n) {
   while (n-- > 0) {
      printf("+ ");
      f(n-1); } }
```

   ◯ Α. `+ + + + + +`
   ◯ Β. `+ + + + + + +`
   ◯ Γ. `+ + + + + + + +`

19. Για να είναι ένας ακέραιος N μεγαλύτερος του 2 πρώτος, αρκεί να ...
◯ A. ... μην διαιρείται με κανέναν από τους αριθμούς 2, 3, 5 ή 7
◯ B. ... είναι περιττός και να μην έχει διαιρέτη μεγαλύτερο του 1 και μικρότερο του $\sqrt{N}$
◯ Γ. ... μην έχει διαιρέτη μεγαλύτερο του 1 και μικρότερο ή ίσο του $N/2$

20. Για να εκτυπώσουμε ταξινομημένους τους κόμβους ενός ταξινομημένου δυαδικού δέντρου, αρκεί να ...
◯ A. ... εκτυπώσουμε πρώτα τη ρίζα του δοθέντος δέντρου, μετά να εκτυπώσουμε αναδρομικά τους κόμβους του αριστερού υποδέντρου της ρίζας και, τέλος, να εκτυπώσουμε αναδρομικά τους κόμβους του δεξιού υποδέντρου της ρίζας
◯ B. ... εκτυπώσουμε πρώτα αναδρομικά τους κόμβους του αριστερού υποδέντρου της ρίζας, μετά να εκτυπώσουμε τη ρίζα του δοθέντος δέντρου και, τέλος, να εκτυπώσουμε αναδρομικά τους κόμβους του δεξιού υποδέντρου της ρίζας
◯ Γ. ... εκτυπώσουμε πρώτα αναδρομικά τους κόμβους του αριστερού υποδέντρου της ρίζας, μετά να εκτυπώσουμε αναδρομικά τους κόμβους του δεξιού υποδέντρου της ρίζας και, τέλος, να εκτυπώσουμε τη ρίζα του δοθέντος δέντρου

21. Πότε μπορούμε να κάνουμε δυαδική αναζήτηση σε ένα πίνακα;
◯ A. Όταν ο πίνακας είναι ταξινομημένος
◯ B. Όταν ο πίνακας περιέχει 2^K στοιχεία, για κάποιο K θετικό ακέραιο
◯ Γ. Όταν τα στοιχεία του πίνακα είναι τύπου `int` ή `double`

22. Ποια είναι η πολυπλοκότητα του καλύτερου αλγορίθμου για να υπολογίσουμε το πλήθος των στοιχείων μιας απλά συνδεδεμένης λίστας με n στοιχεία;
◯ A. $O(n)$
◯ B. $O(n^2)$
◯ Γ. $O(logn)$

23. Αν θέλαμε να ταξινομήσουμε ένα πίνακα με τη γρήγορη μέθοδο ταξινόμησης, πότε θα αναμέναμε να έχει η μέθοδος αυτή τη χειρότερη απόδοση;
◯ A. Όταν τα στοιχεία του πίνακα είναι ακέραιοι αριθμοί σε εντελώς τυχαία σειρά
◯ B. Όταν τα στοιχεία του πίνακα είναι αριθμοί κινητής υποδιαστολής σε εντελώς τυχαία σειρά
◯ Γ. Όταν τα στοιχεία του πίνακα είναι οποιουδήποτε τύπου, ήδη ταξινομημένα σε αύξουσα σειρά

24. Μία εύλογη επέκταση των δυαδικών δέντρων είναι τα τριαδικά δέντρα, δηλαδή δέντρα που κάθε κόμβος μπορεί να έχει μέχρι και τρεις απογόνους. Ποιο είναι το μέγιστο πλήθος κόμβων που μπορεί να έχει ένα τριαδικό δέντρο με d επίπεδα;
◯ A. 3^d
◯ B. $(3^d - 1)/2$
◯ Γ. 3^{2d-1}

25. Από ποια εργασία του μαθήματος θα μπορούσε να προέρχεται το παρακάτω απόσπασμα κώδικα;

```
.................................
int ch, codepoint;
while ((ch = getchar()) != EOF) {
   if ((ch & 0x80) == 0) {
      codepoint = ch;
   }
   else {
      if ((ch & 0xE0) == 0xC0) {
         codepoint = (ch & 0x1F);
         if ((ch = getchar()) == EOF) {
            printf("Unexpected EOF\n");
            return 1;
         }
.................................
```

◯ A. Πρώτη εργασία
◯ B. Δεύτερη εργασία
◯ Γ. Τρίτη εργασία

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Γράψτε ένα πρόγραμμα C το οποίο να δέχεται στη γραμμή εντολής ένα θετικό ακέραιο N και να βρίσκει όλους τους αριθμούς που είναι μικρότεροι ή ίσοι του N και οι οποίοι να μπορούν να γραφούν σαν άθροισμα τετραγώνων τεσσάρων διαδοχικών πρώτων αριθμών. Σημειώστε ότι **δεν επιτρέπεται να χρησιμοποιήσετε πραγματικούς αριθμούς, συναρτήσεις της μαθηματικής βιβλιοθήκης και πίνακες** (εξαιρείται το όρισμα `argv[]` της `main()`). Μία ενδεικτική εκτέλεση:

```
$ ./4consprsq 5000
 2^2 +  3^2 +  5^2 +  7^2 =   87
 3^2 +  5^2 +  7^2 + 11^2 =  204
 5^2 +  7^2 + 11^2 + 13^2 =  364
 7^2 + 11^2 + 13^2 + 17^2 =  628
11^2 + 13^2 + 17^2 + 19^2 =  940
13^2 + 17^2 + 19^2 + 23^2 = 1348
17^2 + 19^2 + 23^2 + 29^2 = 2020        // interesting, eh?
19^2 + 23^2 + 29^2 + 31^2 = 2692
23^2 + 29^2 + 31^2 + 37^2 = 3700
29^2 + 31^2 + 37^2 + 41^2 = 4852
```

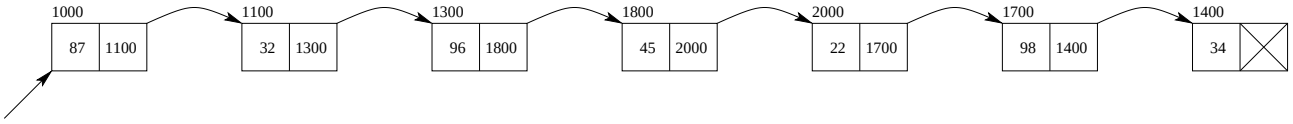
Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Δίνεται η παρακάτω συνάρτηση C, καθώς και ο συνήθης ορισμός ενός κόμβου λίστας ακεραίων:

```
Listptr spl(Listptr list)
{  Listptr fp = list, prev;
   while (fp != NULL) {
      fp = fp->next;
      prev = list; list = list->next;
      if (fp != NULL) fp = fp->next; }
   if (list != NULL) prev->next = NULL;
   return list; }
```

```
typedef struct listnode *Listptr;

struct listnode {
  int value;
  Listptr next;
};
```

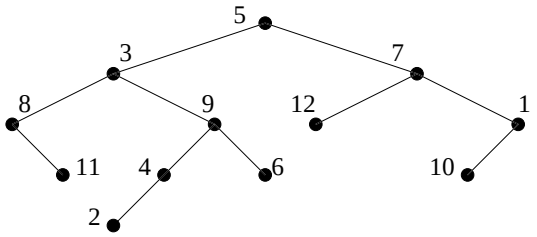
Αν η συνάρτηση `spl()` κληθεί με όρισμα τη διεύθυνση του πρώτου κόμβου της λίστας που φαίνεται στη συνέχεια, δηλαδή το 1000, τι θα επιστρέψει στο όνομά της και τι αλλαγές θα έχουν γίνει ενδεχομένως στη λίστα που της δόθηκε; Δείξτε διαδοχικά τις τιμές που θα παίρνουν οι μεταβλητές της συνάρτησης και σχεδιάστε την τελική κατάσταση της μνήμης μετά την κλήση της συνάρτησης. Περιγράψτε **σε τρεις γραμμές το πολύ** τη λειτουργικότητα της συνάρτησης.



Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Στη C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου ακεραίων ως εξής:

```
typedef struct treenode *Treeptr;

struct treenode {
  int v;
  Treeptr left;
  Treeptr right;
};
```



Γράψτε μία συνάρτηση C που θα δέχεται ως όρισμα ένα δυαδικό δέντρο, δηλαδή τη διεύθυνση του κόμβου-ρίζα τύπου `Treeptr`, και θα εκτυπώνει τα περιεχόμενα των κόμβων του δέντρου κατά επίπεδα, από επάνω προς τα κάτω, και, για κάθε επίπεδο, από αριστερά προς τα δεξιά. Για παράδειγμα, η συνάρτηση θα πρέπει για το παραπάνω δέντρο να εκτυπώσει:

```
5 3 7 8 9 12 1 11 4 6 10 2
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Γραπτή εξέταση Α΄ Περιόδου 2023 (23/1/2023)

Διάρκεια εξέτασης: 120 λεπτά

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (55/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση και **σημειώστε την επάνω στο έντυπο αυτό**. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό. Η "μη απάντηση" δεν βαθμολογείται, ούτε θετικά, ούτε αρνητικά. Σε περίπτωση αλλαγής της αρχικής επιλογής σας σε κάποια ερώτηση, θα πρέπει η τελική απάντησή σας να είναι απόλυτα σαφής, διαφορετικά θα θεωρηθεί ως "μη απάντηση".

1. Τις γραμμές ενός προγράμματος C που ξεκινούν με τον χαρακτήρα # τις διαχειρίζεται ο ...
   ◯ Α. ... προεπεξεργαστής
   ◯ Β. ... μεταγλωττιστής
   ◯ Γ. ... συνδέτης

2. Πριν την κλήση μίας συνάρτησης σ΄ ένα πηγαίο πρόγραμμα C, πρέπει να έχει προηγηθεί ο πλήρης ορισμός της συνάρτησης, ή απλώς το πρωτότυπό της. Ποιο σύστημα χρειάζεται αυτή τη γνώση για να λειτουργήσει σωστά;
   ◯ Α. Ο προεπεξεργαστής
   ◯ Β. Ο μεταγλωττιστής
   ◯ Γ. Ο συνδέτης

3. Ποια είναι η τιμή της αριθμητικής παράστασης `(5 * (22 / 5)) / 10.0` στην C;
   ◯ Α. 2.0
   ◯ Β. 2.2
   ◯ Γ. 2

4. Αν οι μεταβλητές x, y και z είναι τύπου `unsigned char`, ποια τιμή θα πάρει η x μετά την εκτέλεση της ακολουθίας εντολών `y = 011; z = 0x7C; x = ~((y >> 1) ^ z);` στην C;
   ◯ Α. 255
   ◯ Β. 134
   ◯ Γ. 135

5. Η σταθερά χαρακτήρα `'e'` στην C ισούται με ...
   ◯ Α. ... τη συμβολοσειρά `"e"`
   ◯ Β. ... τη συμβολοσειρά `"c+2"`
   ◯ Γ. ... την τιμή της παράστασης `'c'+2`

6. Για τις τοπικές/αυτόματες μεταβλητές μίας συνάρτησης σ΄ ένα πρόγραμμα C δεσμεύεται χώρος στη στοίβα ...
   ◯ Α. ... κατά τη μεταγλώττιση του προγράμματος
   ◯ Β. ... μόλις αρχίσει η εκτέλεση του προγράμματος
   ◯ Γ. ... κάθε φορά που καλείται η συνάρτηση

7. Η εντολή `y = (x+1)++;` στην C είναι ...
   ◯ Α. ... συντακτικά λανθασμένη
   ◯ Β. ... ισοδύναμη με την ακολουθία εντολών `y = x++; ++y;`
   ◯ Γ. ... ισοδύναμη με την ακολουθία εντολών `x++; y = x+1;`

8. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int *p, x[] = {5, 7, 6, 2, 6, 1, 0, 4};
p = x;
do {
   p += 2;
} while (*(p-2) = *p);
printf("%d %d\n", p[0], p[1]);
```

   ◯ Α. `6 1`
   ◯ Β. `0 4`
   ◯ Γ. `5 7`

9. Με ποιον τρόπο πρέπει να έχει δηλωθεί η μεταβλητή `fun` στην C, ώστε η εντολή `fun = pow`, όπου `pow` είναι η συνάρτηση της μαθηματικής βιβλιοθήκης για την ύψωση σε δύναμη, να είναι συνταχτικά ορθή;
   ◯ Α. `double *fun(double, double);`
   ◯ Β. `double (*fun)(double, double);`
   ◯ Γ. `double fun(double);`

10. Τι έχετε να σχολιάσετε για το παρακάτω τμήμα κώδικα C;

```
char *mess = "reform";
mess[0] = 'd';
printf("%s\n", mess);
```

   ◯ Α. Θα προκύψει σφάλμα κατά τη μεταγλώττιση
   ◯ Β. Θα μεταγλωττιστεί επιτυχώς, αλλά θα προκύψει σφάλμα κατά την εκτέλεση
   ◯ Γ. Θα μεταγλωττιστεί επιτυχώς και κατά την εκτέλεση θα εκτυπωθεί το `deform`

11. Τι έχετε να σχολιάσετε για το παρακάτω τμήμα κώδικα C;

```
char mess[] = "reform";
mess += 2;
printf("%s\n", mess);
```

   ◯ Α. Θα προκύψει σφάλμα κατά τη μεταγλώττιση
   ◯ Β. Θα μεταγλωττιστεί επιτυχώς, αλλά θα προκύψει σφάλμα κατά την εκτέλεση
   ◯ Γ. Θα μεταγλωττιστεί επιτυχώς και κατά την εκτέλεση θα εκτυπωθεί το `form`

12. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
#define tentimes(A) (10*A)
printf("%d\n", 60-tentimes(4+2));
```

   ◯ Α. 0
   ◯ Β. 18
   ◯ Γ. 22

13. Αν σ΄ ένα πρόγραμμα C, για τις μεταβλητές `x` και `p` έχει προηγηθεί η δήλωση `int x, *p;` , τι μπορείτε να σχολιάσετε για τις εντολές `*p = x;` και `p = &x;` ;
   ◯ Α. Είναι συνταχτικά σωστές και ταυτόσημες
   ◯ Β. Είναι συνταχτικά σωστές, αλλά δεν είναι ταυτόσημες
   ◯ Γ. Τουλάχιστον μία από αυτές είναι συνταχτικά λανθασμένη

14. Η εντολή `&x = p;` στην C ...
   ◯ Α. ... αναθέτει στον δείκτη `x` τη διεύθυνση της μεταβλητής `p`
   ◯ Β. ... αναθέτει στη μεταβλητή `x` την τιμή του δείκτη `p`
   ◯ Γ. ... είναι συνταχτικά λανθασμένη

15. Αν σ΄ ένα πρόγραμμα C, οι μεταβλητές `x` και `y` έχουν δηλωθεί ως `int x, y;` , τότε η εντολή `y = x; ...`
   ◯ Α. ... είναι ισοδύναμη με την εντολή `y = *&x;`
   ◯ Β. ... είναι ισοδύναμη με την εντολή `y = &*x;`
   ◯ Γ. ... δεν είναι ισοδύναμη με καμία από τις εντολές `y = *&x;` ή `y = &*x;`

16. Αν η συνάρτηση `f()` είναι ορισμένη στην C ως

```
void f(int x)
{  if ((x-3) && (x-4)) {
       f(x+1);
       f(x+2);
   }
}
```

   και κάνουμε την κλήση `f(0)`, πόσες φορές θα κληθεί τελικά η `f(3)`;
   ◯ Α. 2
   ◯ Β. 3
   ◯ Γ. Υπερβολικά πολλές, με αποτέλεσμα να προκληθεί υπερχείλιση της στοίβας

17. Η μέθοδος ταξινόμησης του σωρού ...
    ◯ Α. ... είναι επαναληπτική
    ◯ Β. ... είναι αναδρομική
    ◯ Γ. ... απαιτεί δυναμικά δεσμευμένο πίνακα για να λειτουργήσει σωστά

18. Τι έχετε να σχολιάσετε για το παρακάτω τμήμα κώδικα C;

```
char *s1, *s2 = "careta";
strcpy(s1, s2);
printf("%s-%s\n", s1, s2);
```

    ◯ Α. Θα προκύψει σφάλμα κατά τη μεταγλώττιση
    ◯ Β. Θα μεταγλωττιστεί επιτυχώς, αλλά πιθανότατα θα προκύψει σφάλμα κατά την εκτέλεση
    ◯ Γ. Θα μεταγλωττιστεί επιτυχώς και κατά την εκτέλεση θα εκτυπωθεί το `careta-careta`

19. Ποια είναι η πολυπλοκότητα χρόνου του καλύτερου αλγορίθμου για να ελέγξουμε αν οι κόμβοι μίας απλά συνδεδεμένης λίστας N ακεραίων είναι ταξινομημένοι σε αύξουσα σειρά;
    ◯ Α. $O(N)$
    ◯ Β. $O(N^2)$
    ◯ Γ. $O(N \cdot logN)$

20. Ποια είναι η πολυπλοκότητα χρόνου του καλύτερου αλγορίθμου που μπορούμε να γράψουμε για να βρούμε το πλήθος των διαιρετών ενός ακεραίου αριθμού N;
    ◯ Α. $O(N)$
    ◯ Β. $O(logN)$
    ◯ Γ. $O(\sqrt{N})$

21. Η μνήμη που απαιτείται για τη φύλαξη ενός κόμβου-φύλλου ενός δυαδικού δέντρου είναι λιγότερη από την απαιτούμενη μνήμη για τη φύλαξη ενός ενδιάμεσου κόμβου. Τι γνώμη έχετε;
    ◯ Α. Ισχύει πάντοτε
    ◯ Β. Ισχύει εκτός εάν πρόκειται για δέντρο με κόμβους που είναι συμβολοσειρές
    ◯ Γ. Δεν ισχύει γιατί η απαιτούμενη μνήμη είναι ίδια για κάθε κόμβο του δέντρου

22. Η πολυπλοκότητα χρόνου της μεθόδου ταξινόμησης της συγχώνευσης είναι ίδια με την πολυπλοκότητα χρόνου της μεθόδου ταξινόμησης του σωρού. Τι γνώμη έχετε;
    ◯ Α. Ισχύει
    ◯ Β. Η μέθοδος της συγχώνευσης έχει καλύτερη πολυπλοκότητα από αυτήν του σωρού
    ◯ Γ. Η μέθοδος του σωρού έχει καλύτερη πολυπλοκότητα από αυτήν της συγχώνευσης

23. Η πολυπλοκότητα χρόνου του καλύτερου αλγορίθμου για να υπολογίσουμε το άθροισμα των στοιχείων μίας απλά συνδεδεμένης λίστας N ακεραίων είναι $O(N)$ ...
    ◯ Α. ... όταν ο αλγόριθμος υλοποιηθεί επαναληπτικά
    ◯ Β. ... όταν ο αλγόριθμος υλοποιηθεί αναδρομικά
    ◯ Γ. ... είτε με επαναληπτική είτε με αναδρομική υλοποίηση του αλγορίθμου

24. Πόσα διαφορετικά ταξινομημένα δυαδικά δέντρα υπάρχουν με τρεις κόμβους που έχουν τις τιμές 2, 5 και 8;
    ◯ Α. 1
    ◯ Β. 5
    ◯ Γ. 6

25. Το 2023 είναι ...
    ◯ Α. ... πρώτος αριθμός[1]
    ◯ Β. ... ελεύθερος-τετραγώνου αριθμός[2]
    ◯ Γ. ... ελαττωματικός αριθμός[3]

26. Ποια είναι η εκτίμησή σας για τα συνολικά μόρια που πήρατε από τις προηγούμενες 25 ερωτήσεις; Δεν υπάρχει αρνητική βαθμολογία στην ερώτηση αυτή και η σωστή απάντηση επιβραβεύεται με 5 μόρια επιπλέον.
    ◯ Α. 31–50
    ◯ Β. 15–30
    ◯ Γ. Λιγότερα από 15

---

[1] Έχει ως μόνους διαιρέτες το 1 και τον εαυτό του.

[2] Στην ανάλυσή του σε γινόμενο πρώτων παραγόντων, δεν εμφανίζεται κάποιος παράγοντας περισσότερες από μία φορά.

[3] Το άθροισμα των διαιρετών του, εξαιρουμένου του εαυτού του, είναι μικρότερο από τον ίδιο τον αριθμό.

Θέμα 2 (30/100):
Οι αριθμοί *αρμονικών διαιρετών* (ή αριθμοί Ore, προς τιμήν του Νορβηγού μαθηματικού, που τους όρισε το 1948) είναι εκείνοι οι θετικοί ακέραιοι για τους οποίους ισχύει ότι ο μέσος αρμονικός των διαιρετών τους (συμπεριλαμβανομένων του 1 και του εαυτού τους) είναι ακέραιος αριθμός. Δηλαδή, ένας αριθμός είναι Ore όταν έχει k διαιρέτες, τους d_1, d_2, ..., d_k, και ισχύει ότι το κλάσμα

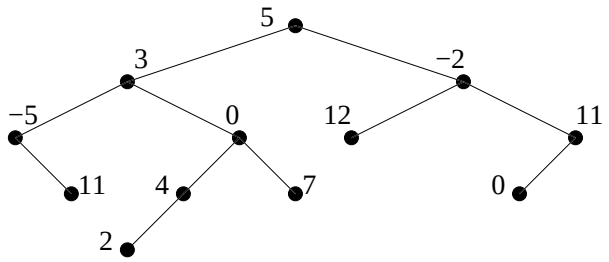
$$\frac{k}{\frac{1}{d_1}+\frac{1}{d_2}+\ldots+\frac{1}{d_k}}$$

είναι ακέραιος αριθμός. Για παράδειγμα, το 6, που έχει διαιρέτες τους 1, 2, 3 και 6, είναι αριθμός Ore, αφού $\frac{4}{\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{6}}=2$. Ομοίως και το 140, αφού $\frac{12}{\frac{1}{1}+\frac{1}{2}+\frac{1}{4}+\frac{1}{5}+\frac{1}{7}+\frac{1}{10}+\frac{1}{14}+\frac{1}{20}+\frac{1}{28}+\frac{1}{35}+\frac{1}{70}+\frac{1}{140}}=5$. Γράψτε ένα πρόγραμμα C το οποίο να καλείται με έναν ακέραιο N σαν όρισμα στη γραμμή εντολής και να εκτυπώνει όλους τους αριθμούς Ore που είναι μικρότεροι ή ίσοι του N. **Δεν επιτρέπεται να χρησιμοποιήσετε πίνακες** (εξαιρείται το όρισμα `argv[]` της `main`). Επίσης, **η πολυπλοκότητα χρόνου του αλγορίθμου που θα υλοποιηθεί πρέπει να είναι καλύτερη από $O(N^2)$ (50% της βαθμολογίας του θέματος)**. Τέλος, υπενθυμίζεται ότι **όλοι σχεδόν οι πραγματικοί αριθμοί δεν μπορούν να αναπαρασταθούν με απόλυτη ακρίβεια στον υπολογιστή**, συνεπώς απαντήσεις που χρησιμοποιούν αριθμητική σε πραγματικούς αριθμούς είναι πιθανότατα εξ αρχής λάθος. Μία ενδεικτική εκτέλεση του προγράμματος είναι η εξής:

```
$ ./ore 50000
1 6 28 140 270 496 672 1638 2970 6200 8128 8190 18600 18620 27846 30240 32760
```

Θέμα 3 (20/100):
Στην C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου ακεραίων ως εξής:

```
typedef struct treenode *Treeptr;

struct treenode {
  int value;
  Treeptr left;
  Treeptr right;
};
```



Υλοποιήστε μία συνάρτηση C με πρωτότυπο `int pathssum(Treeptr, int)`, η οποία να παίρνει σαν όρισμα ένα δυαδικό δέντρο ακεραίων και έναν ακέραιο (έστω `sum`) και να επιστρέφει στο όνομά της το πλήθος των μονοπατιών που υπάρχουν στο δέντρο από τη ρίζα μέχρι κάποιο φύλλο, για τα οποία το άθροισμα των περιεχομένων των κόμβων τους ισούται με `sum`. Για παράδειγμα, αν η συνάρτηση αυτή καλείτο για το δέντρο του παραπάνω σχήματος και για `sum` ίσο με 14, θα έπρεπε να επιστρέψει το 3, γιατί υπάρχουν τρία ακριβώς μονοπάτια από τη ρίζα σε φύλλο που τα περιεχόμενα των κόμβων τους έχουν άθροισμα 14, τα 5/3/-5/11, 5/3/0/4/2 και 5/-2/11/0.

Θέμα 4 (25/100):
Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`bigmod.c`"), που να δέχεται στη γραμμή εντολής τρεις θετικούς ακεραίους B, E και M και το οποίο να υπολογίζει και να εκτυπώνει το $B^E \bmod M$ (υπόλοιπο της διαίρεσης του B^E με το M). Το πρόγραμμα που θα γράψετε θα πρέπει να δουλεύει σωστά και αποδοτικά, ακόμα και για μεγάλες τιμές του E, όπως φαίνεται στα παρακάτω παραδείγματα, διαφορετικά δεν θα βαθμολογηθεί. Υπενθυμίζεται ότι οι ακέραιοι στην C, ακόμα και οι `unsigned long long`, δεν μπορούν να πάρουν τιμές μεγαλύτερες του 2^{64} ($< 2 \cdot 10^{19}$). Παραδείγματα εκτέλεσης του προγράμματος:

```
$ ./bigmod 123 4567 89
55
$ ./bigmod 1000000 1000000 592
112
$ ./bigmod 7 2000000000 12345
1276
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2014 (28/3/2014)

1. (α΄) Τι κερδίζουμε και τι χάνουμε στη C όταν μία ακέραια μεταβλητή τη δηλώσουμε με τον προσδιοριστή **unsigned**; Σε ποιες περιπτώσεις χρησιμοποιούμε τον προσδιοριστή **static** στη δήλωση μίας μεταβλητής στη C;

   (β΄) Όταν εκτελεσθεί το πρόγραμμα C του διπλανού σχήματος σε κάποιον συνηθισμένο υπολογιστή, δίνει το εξής αποτέλεσμα:

   ```
   1 2 6 24 120 720 5040 -25216
   ```

   Εξηγήστε σε μία πρόταση τη λειτουργικότητα αυτού του προγράμματος και δικαιολογήστε τα αποτελέσματά του.

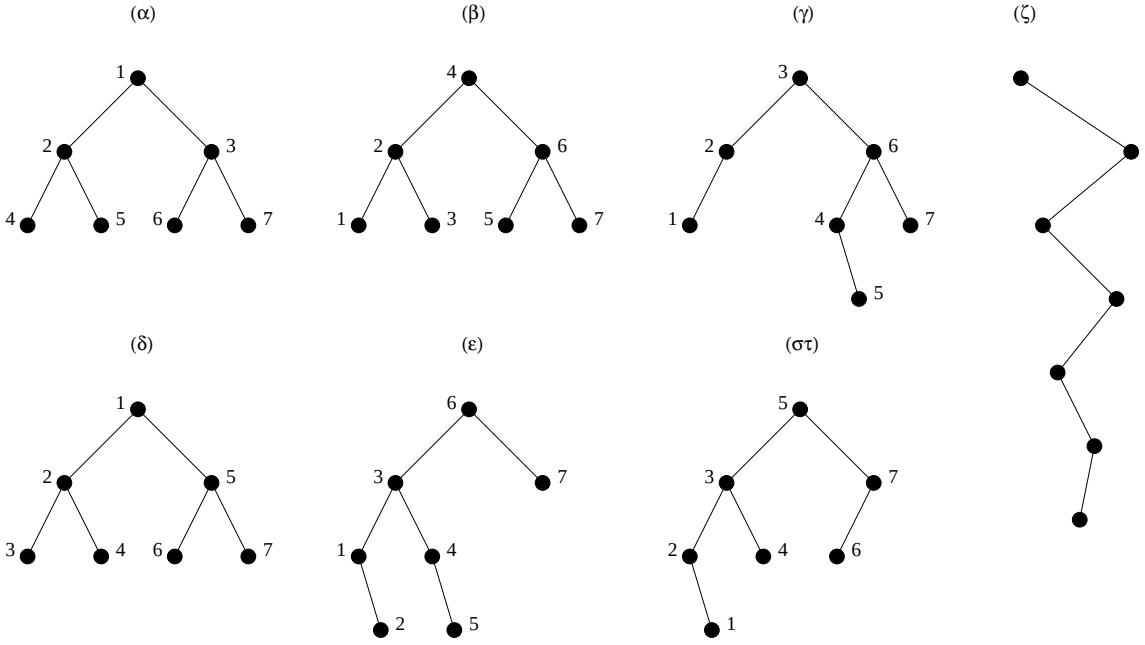
```
#include <stdio.h>

int main(void)
{  short int n, p, i;
   for (n = 1 ; n < 9 ; n++) {
      p = 1;
      for (i = 1 ; i <= n ; i++)
         p = p * i;
      printf("%d ", p);
   }
   printf("\n");
   return 0;
}
```

   (γ΄) Δεδομένου ότι οι μεταβλητές `x` και `y` είναι δηλωμένες ως ακέραιες, ποια είναι η λειτουργικότητα της παρακάτω ακολουθίας εντολών;

   ```
   x = x^y ; y = x^y ; x = x^y ;
   ```

   Δικαιολογήστε την άποψή σας μέσω ενός κατάλληλου παραδείγματος. Υπενθυμίζεται ότι ο τελεστής ^ της C είναι του bitwise XOR (αποκλειστικό Ή).

2. (α΄) Ποια από τα παρακάτω δυαδικά δέντρα (α), (β), (γ), (δ), (ε) και (στ) είναι ταξινομημένα; Τοποθετήστε κατάλληλα τους ακέραιους αριθμούς από το 1 έως το 7 στους κόμβους του δυαδικού δέντρου (ζ) ώστε να είναι ταξινομημένο.

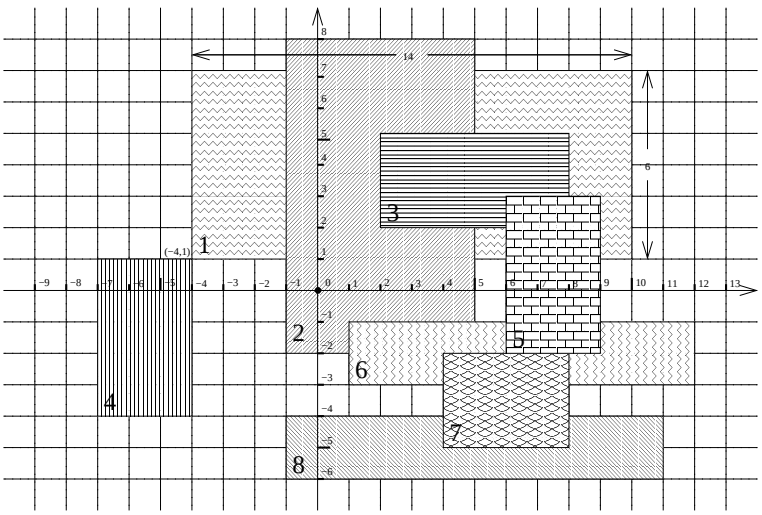


   (β΄) Ορίστε μία συνάρτηση C με πρωτότυπο `int ordtree(Treeptr)`, η οποία να επιστρέφει στο όνομά της 1, αν το δυαδικό δέντρο ακεραίων που της δίνεται σαν όρισμα είναι ταξινομημένο, ή 0, σε αντίθετη περίπτωση. Η συνάρτηση που θα γράψετε θα πρέπει να είναι αποδοτική ακόμα και για ιδιαίτερα μεγάλα δέντρα. Το ιδανικό είναι να έχει πολυπλοκότητα $O(n)$, όπου n είναι το πλήθος των κόμβων του δέντρου. Υπενθυμίζεται ότι το `Treeptr` είναι ο τύπος δείκτη σε κόμβο δυαδικού δέντρου ακεραίων που γνωρίζετε από το μάθημα.

3. Ένας ακέραιος μεγαλύτερος του 1 λέγεται ότι είναι *παλινδρομικός πρώτος* (palindromic prime), όταν είναι πρώτος (έχει σαν μόνους διαιρέτες τον εαυτό του και το 1) και όταν αντιστρέψουμε τη σειρά των ψηφίων του, προκύπτει ο ίδιος αριθμός. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`palinprimes.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (stdin) ένα θετικό ακέραιο N και να εκτυπώνει όλους τους παλινδρομικούς πρώτους που είναι μικρότεροι ή ίσοι του N. Μία ενδεικτική εκτέλεση του προγράμματος είναι η εξής:

```
% ./palinprimes
Please, give maximum number: 10600
2 3 5 7 11 101 131 151 181 191 313 353 373 383 727 757 787 797 919 929 10301 10501
```

4. Έστω ότι είναι δεδομένα N ορθογώνια παραλληλόγραμμα τοποθετημένα σε ένα ορθογώνιο σύστημα συντεταγμένων, με τις πλευρές τους παράλληλες στους άξονες, όπως φαίνεται στο διπλανό σχήμα. Κάθε παραλληλόγραμμο καθορίζεται πλήρως από τις συντεταγμένες της κάτω αριστερής κορυφής του, το πλάτος του και το ύψος του. Μπορείτε να θεωρήσετε ότι οι συντεταγμένες των κορυφών του παραλληλογράμμου, το πλάτος του και το ύψος του είναι ακέραιοι αριθμοί. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`overlrects.c`"), το οποίο αρχικά να δέχεται από τη γραμμή εντολής έναν ακέραιο N (πλήθος ορθογωνίων παραλληλογράμμων). Στη συνέχεια, το πρόγραμμα να διαβάζει από την πρότυπη είσοδο (stdin) τα δεδομένα για N ορθογώνια παραλληλόγραμμα, δηλαδή, για καθένα από αυτά, τις συντεταγμένες (x, y) της κάτω αριστερής κορυφής του, το πλάτος του w και το ύψος του h. Το πρόγραμμα να εκτυπώνει όλα τα ζευγάρια παραλληλογράμμων που επικαλύπτονται, καθώς και το εμβαδόν του κοινού τους τμήματος. Ένα παράδειγμα εκτέλεσης φαίνεται στη συνέχεια:




```
% ./overlrects 8
-4  1 14  6
-1 -2  6 10
 2  2  6  3
-7 -4  3  5
 6 -2  3  5
 1 -3 11  2
 4 -5  4  3
-1 -6 12  2
Rectangles 1 and 2 overlap for 36 square units
Rectangles 1 and 3 overlap for 18 square units
Rectangles 1 and 5 overlap for 6 square units
Rectangles 2 and 3 overlap for 9 square units
Rectangles 2 and 6 overlap for 4 square units
Rectangles 3 and 5 overlap for 2 square units
Rectangles 5 and 6 overlap for 3 square units
Rectangles 6 and 7 overlap for 4 square units
Rectangles 7 and 8 overlap for 4 square units
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2007 (8/10/2007)

1. Η ακολουθία Tribonacci ορίζεται ως εξής:

$$a_n = \begin{cases} 0 & \text{αν } n = 0 \\ 1 & \text{αν } n = 1 \quad \text{ή} \quad n = 2 \\ a_{n-1} + a_{n-2} + a_{n-3} & \text{αν } n \geq 3 \end{cases}$$

Γράψτε σε C δύο συναρτήσεις, έστω τις `int tribrec(int n)` και `int tribiter(int n)`, οι οποίες, για δεδομένο `n`, να υπολογίζουν και να επιστρέφουν στο όνομά τους τον όρο a_n της ακολουθίας Tribonacci, η μεν πρώτη συνάρτηση μέσω **αναδρομής**, η δε δεύτερη με **επαναληπτικό** τρόπο. Δεν χρειάζεται να γράψετε `main` συνάρτηση που να καλεί τις συναρτήσεις αυτές. Τέλος, δώστε την εκτίμησή σας για την αποδοτικότητα των συναρτήσεων αυτών, ιδιαίτερα για μεγάλες τιμές του `n`, και δικαιολογήστε την απάντησή σας.

2. Αιτιολογήστε, *σε 15 το πολύ γραμμές*, την ορθότητα ή μη των παρακάτω δηλώσεων, σχετικά με το τι ισχύει στη γλώσσα προγραμματισμού C:

   (α΄) Είναι ακριβώς το ίδιο να χρησιμοποιούμε αντί για το `'n'`, είτε το `'\n'` είτε το `"n"`.

   (β΄) Οι παρακάτω εντολές είναι απολύτως ισοδύναμες:

```
char str[] = "computer";
char *str = "computer";
char str[8] = {'c', 'o', 'm', 'p', 'u', 't', 'e', 'r'};
char str[] = {'c', 'o', 'm', 'p', 'u', 't', 'e', 'r', '\0'};
```

   (γ΄) Οι παρακάτω δηλώσεις είναι απολύτως ισοδύναμες:

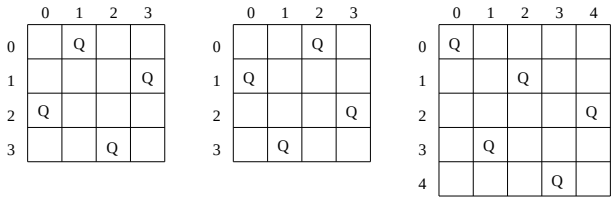
```
char **fun(char *);
char *(*fun)(char *);
char (**fun)(char *);
```

3. Το πρόβλημα των 8 βασιλισσών συνίσταται στην τοποθέτηση 8 βασιλισσών σε μία σκακιέρα (πλαίσιο 8×8), έτσι ώστε αυτές οι βασίλισσες να μην απειλούνται μεταξύ τους, με βάση τους κανόνες που ισχύουν στο σκάκι. Αυτό σημαίνει ότι πρέπει να τοποθετηθούν έτσι ώστε να μην υπάρχουν δύο βασίλισσες στην ίδια γραμμή, ούτε στην ίδια στήλη, ούτε στην ίδια διαγώνιο. Καταλαβαίνετε ότι το πρόβλημα αυτό γενικεύεται εύκολα και για N βασίλισσες, οι οποίες θα πρέπει να τοποθετηθούν σ΄ ένα πλαίσιο $N \times N$ έτσι ώστε να μην απειλούνται μεταξύ τους.

   Ένας τρόπος επίλυσης του προβλήματος των N βασιλισσών είναι να τοποθετούμε μία βασίλισσα σε κάθε γραμμή, με όλους τους δυνατούς τρόπους, και για όλες τις πιθανές τοποθετήσεις N βασιλισσών (συνολικού πλήθους N^N) να ελέγχουμε αν ισχύει ο περιορισμός της "μη απειλής". Η αναπαράσταση κάθε λύσης μπορεί να γίνει με N αριθμούς (από 0 έως $N-1$), έναν για κάθε γραμμή, όπου ο καθένας από τους αριθμούς αυτούς αντιστοιχεί στη στήλη στην οποία βρίσκεται η βασίλισσα στη συγκεκριμένη γραμμή. Ενδεικτικές εκτελέσεις ενός προγράμματος C (έστω με όνομα `queens.c`) που εφαρμόζει αυτήν τη μέθοδο φαίνονται παρακάτω (κάθε σειρά στα αποτελέσματα είναι μία λύση του προβλήματος):

```
% ./queens 4
1 3 0 2
2 0 3 1
% ./queens 5
0 2 4 1 3
0 3 1 4 2
.................
```

Οι λύσεις του προβλήματος των 4 βασιλισσών και η πρώτη από τις λύσεις του προβλήματος των 5 βασιλισσών που βρέθηκαν από το πρόγραμμα φαίνονται στα σχήματα δεξιά. Μία πιθανή υλοποίηση του `queens.c`, ημιτελής όμως, που εφαρμόζει τον αλγόριθμο που περιγράφηκε, δίνεται στη συνέχεια. Συμπληρώστε τον κώδικα όπου υπάρχουν αποσιωπητικά και εξηγήστε τη λειτουργία του προγράμματος στα, κατά την κρίση σας, "δύσκολα" σημεία του (κυρίως στις συναρτήσεις `check` και `generate`).

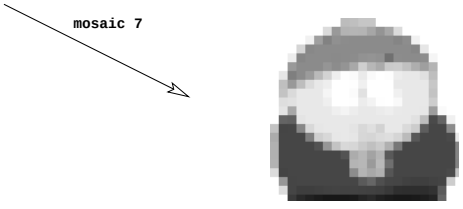



```
int check(int n, int *p)
{  int i, j;
   for (i = 0 ; i < .... ; i++)
      for (j = .... ; .... ; j++)
         if ( .................. )
           return 0;
   return 1; }

int generate(int n, int *p)
{  int k, r;
   k = n;
   do {
      k--;
      p[k] = (p[k]+1) .... n;
   } while ( ................... );
   r = ..................... ;
   return r; }
```

```
void print(int n, int *p)
{  int i;
   for (i = 0 ; i < n ; i++)
      printf("%d ", p[i]);
   printf("\n"); }

main(int argc, char *argv[])
{  int i, *p, n = 8;
   if (argc > 1) n = atoi(argv[1]);
   p = malloc(n * sizeof(int));
   if (p == NULL) return 1;
   for (i = 0 ; i < n ; i++)
     p[i] = 0;
   do {
     if (check(n, p))
       print(n, p);
   } while (generate(n, p)); }
```

4. Πιθανώς γνωρίζετε ότι μία ασπρόμαυρη εικόνα, αλλά με αποχρώσεις του γκρι, μπορεί να αναπαρασταθεί με ένα διδιάστατο πίνακα, όπου η τιμή κάθε στοιχείου του, έστω από 0 (μαύρο) έως 255 (άσπρο), είναι το επίπεδο μαύρου-άσπρου του αντίστοιχου εικονοστοιχείου (pixel) της εικόνας. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`mosaic.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (`stdin`) μία εικόνα, πρώτα τις διαστάσεις της (ύψος και πλάτος) και στη συνέχεια τον πίνακα των εικονοστοιχείων της, και να εκτυπώνει στην πρότυπη έξοδο (`stdout`) την εικόνα σε "στυλ μωσαϊκού" με πλακίδια πλευράς d, το οποίο δίνεται σαν όρισμα στη γραμμή εντολής. Όλα τα εικονοστοιχεία ενός πλακιδίου του μωσαϊκού στην τελική εικόνα έχουν το ίδιο χρώμα, η τιμή του οποίου ισούται με το ακέραιο μέρος της μέσης τιμής των τιμών των αντίστοιχων εικονοστοιχείων στην αρχική εικόνα. Δεν είναι απαραίτητο οι διαστάσεις της εικόνας να είναι πολλαπλάσια του d, οπότε στην τελική εικόνα μπορεί να υπάρχουν και πλακίδια που δεν είναι τετράγωνα. Παρακάτω, αριστερά είναι ένα παράδειγμα εκτέλεσης και δεξιά το αποτέλεσμα της μετατροπής μίας εικόνας (145×150) σε "στυλ μωσαϊκού".

```
% cat image.txt
5 7
  0   0   0 190   0   0   0
 20  30  40 210 100  50  30
 40  60  80 230 150  90  60
  0   0   0 250   0   0   0
  0   0   0 255   0   0   0
% ./mosaic 3 < image.txt
5 7
 30  30  30 113 113 113  30
 30  30  30 113 113 113  30
 30  30  30 113 113 113  30
  0   0   0  84  84  84   0
  0   0   0  84  84  84   0
```



ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β′ Περιόδου 2006 (21/9/2006)

1. (α′) Γιατί χρειαζόμαστε πίνακες σε μία γλώσσα προγραμματισμού;

   (β′) Τι εννοούμε με τον όρο "*στατική δέσμευση μνήμης*" για έναν πίνακα; Θα ήταν αρκετό για μία γλώσσα προγραμματισμού να παρείχε τη δυνατότητα μόνο στατικής δέσμευσης για πίνακες; Αν όχι, γιατί;

   (γ′) Τι εννοούμε με τον όρο "*δυναμική δέσμευση μνήμης*" για έναν πίνακα; Με ποιον τρόπο επιλύει η δυναμική δέσμευση τα πιθανά προβλήματα της στατικής δέσμευσης;

   (δ′) Δώστε δύο παραδείγματα προβλημάτων (διατυπωμένα επιγραμματικά σε φυσική γλώσσα) μέσω των οποίων να επιδεικνύεται η επάρκεια της στατικής δέσμευσης και η αναγκαιότητα της δυναμικής δέσμευσης, κατά περίπτωση.

   (ε′) Γράψτε τμήμα προγράμματος C στο οποίο να δεσμεύεται μνήμη για δύο μονοδιάστατους πίνακες ακεραίων `x` και `y`, για τον πρώτο με στατικό τρόπο και για τον δεύτερο με δυναμικό τρόπο.

   (ϛ′) Τι μειονεκτήματα και τι πλεονεκτήματα έχει η χρήση δυναμικά δεσμευμένου πίνακα σε σχέση με τη χρήση συνδεδεμένης λίστας για ένα πρόβλημα;

   Απαντήστε στα προηγούμενα συνοπτικά, κατά προτίμηση *σε μία σελίδα το πολύ*.

2. Στις εξετάσεις κάποιου μαθήματος προγραμματισμού ζητήθηκε από τους φοιτητές να ορίσουν το πρωτότυπο μίας συνάρτησης C, καθώς και να την υλοποιήσουν, η οποία δεδομένων του αθροίσματος και της διαφοράς δύο πραγματικών αριθμών διπλής ακρίβειας, να υπολογίζει και να επιστρέφει αυτούς τους αριθμούς. Επίσης, τους ζητήθηκε να γράψουν και `main` συνάρτηση που να την καλεί και να εκτυπώνει τα αποτελέσματα. Κάποιοι φοιτητές απάντησαν ως εξής:

```
void f(double sum, double diff, double x, double y)
{  x = (sum + diff) / 2;
   y = (sum - diff) / 2; }

main()
{  double x, y, sum = 8.478, diff = 2.312;
   f(sum, diff, x, y);
   printf("%f %f\n", x, y); }
```

   Σχολιάστε την απάντησή τους και βαθμολογήστε την με άριστα το 10. Αν ο βαθμός που βάλατε είναι μεγαλύτερος του 2, τότε δεν έχετε βαθμολογήσει σωστά την απάντηση και θα κριθείτε αρνητικά στο θέμα αυτό! Αν υπάρχει, κατά τη γνώμη σας, πολύ σοβαρό λάθος στην απάντηση αυτών των φοιτητών, προτείνετε, γράφοντας τον κατάλληλο κώδικα C, δύο άλλους τρόπους αντιμετώπισης του προβλήματος, ουσιωδώς διαφορετικούς μεταξύ τους, χωρίς όμως τη χρήση εξωτερικών (καθολικών) μεταβλητών, τους οποίους να μπορούσατε άνετα να βαθμολογήσετε με 10.

3. Το πρόβλημα που καλείσθε να αντιμετωπίσετε σχετίζεται με ναρκοπέδια. Μπορείτε να φανταστείτε ένα ναρκοπέδιο σαν ένα ορθογώνιο πλέγμα διάστασης $n \times m$, δηλαδή σαν ένα πλαίσιο με n γραμμές και m στήλες. Σε κάθε κελί του πλαισίου μπορεί να υπάρχει ή να μην υπάρχει νάρκη.

   Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`mines.c`") το οποίο κατ′ αρχήν να διαβάζει από την πρότυπη είσοδο (`stdin`) τα δεδομένα ενός ναρκοπεδίου. Συγκεκριμένα στην πρώτη γραμμή της εισόδου πρέπει να υπάρχουν οι διαστάσεις του ναρκοπεδίου (n και m) και μετά να ακολουθούν γραμμή-γραμμή τα περιεχόμενα των κελιών, που μπορεί να είναι είτε ο χαρακτήρας "`*`", όταν υπάρχει νάρκη στο κελί, είτε ο χαρακτήρας "`.`", όταν δεν υπάρχει νάρκη. Το πρόγραμμά σας να εκτυπώνει μία τροποποιημένη περιγραφή του ναρκοπεδίου, στην οποία εκεί που υπάρχει νάρκη θα φαίνεται πάλι το "`*`", ενώ στα κελιά που δεν υπάρχει νάρκη θα φαίνεται ένας αριθμός που θα δείχνει

σε πόσα γειτονικά κελιά υπάρχει νάρκη. Σαν γειτονικά θεωρούνται όχι μόνο συνεχόμενα οριζόντια ή κατακόρυφα κελιά, αλλά και συνεχόμενα σε διαγώνια κατεύθυνση. Συνεπώς, ένα κελί μπορεί να έχει μέχρι και 8 γειτονικά κελιά. Το πρόγραμμά σας πρέπει να είναι σε θέση να διαχειριστεί οσοδήποτε μεγάλα ναρκοπέδια, δηλαδή χωρίς κάποιο ρητό περιορισμό στο μέγεθός τους. Μία ενδεικτική εκτέλεση φαίνεται στη συνέχεια:

```
% cat mines.txt
4 9
..*.***..
*...*.*..
....****.
*....**..
% ./mines < mines.txt
12*3***20
*214*8*41
2202****1
*1013**31
%
```

4. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`uncomment.c`") το οποίο να διαβάζει από την πρότυπη είσοδο (`stdin`) ένα πηγαίο πρόγραμμα C και να εκτυπώνει στην πρότυπη έξοδο (`stdout`) το ίδιο πρόγραμμα, έχοντας αφαιρέσει τα σχόλια που περιέχει. Το πρόγραμμα σας να χειρίζεται μόνο τα κλασικά σχόλια `/* ... */` της C, όχι τα `//`. Προσέξτε όμως ότι τα `/*` και `*/` όταν βρίσκονται μέσα σε συμβολοσειρές `"..."` δεν ορίζουν την αρχή ή το τέλος σχολίου. Επίσης, θυμηθείτε ότι αν θέλουμε να βάλουμε μέσα σε μία συμβολοσειρά τον χαρακτήρα ", τον γράφουμε σαν `\"`. Θεωρήστε ότι τα προγράμματα C που θα δίνονται στην είσοδο του προγράμματός σας δεν έχουν συντακτικά λάθη. Τέλος, το πρόγραμμα που θα γράψετε θα πρέπει να δουλεύει σωστά ακόμα και αν δοθεί στην είσοδό του ο πλήρης κώδικας σε C του Unix (το πιάσατε το υποννοούμενο;). Μία ενδεικτική εκτέλεση φαίνεται στη συνέχεια:

```
% cat aprog.c
#include <stdio.h>
/* A "test" program */
main()
{ /* Just a * in
     a comment */
  printf("%d\" - This /* is not */ a comment\n", 12/3);
}
% ./uncomment < aprog.c
#include <stdio.h>

main()
{
  printf("%d\" - This /* is not */ a comment\n", 12/3);
}
%
```

Υπόδειξη: Εκτός από τη γνωστή σας συνάρτηση `int getchar(void)`, που διαβάζει ένα χαρακτήρα από την πρότυπη είσοδο `stdin`, είναι πολύ πιθανό να σας χρησιμεύσει και μία συνάρτηση της πρότυπης βιβλιοθήκης εισόδου/εξόδου, που ίσως δεν γνωρίζετε, η `int ungetc(int char, FILE *stream)`, για την αναίρεση της ανάγνωσης ενός χαρακτήρα από ένα ρεύμα εισόδου. Δηλαδή, αν διαβαστεί το `char` από την πρότυπη είσοδο και διαπιστωθεί ότι κακώς διαβάστηκε και πρέπει να επιστραφεί, ώστε να ξαναδιαβαστεί στη συνέχεια, αυτό επιτυγχάνεται με την κλήση της `ungetc(char, stdin)`.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Α΄ Περιόδου 2007 (27/8/2007)

1. (α΄) Κάθε πρόγραμμα που εκτελείται χρησιμοποιεί, μεταξύ άλλων, και μία στοίβα για τη φύλαξη και διαχείριση δεδομένων. Ποιος είναι ο ρόλος αυτής της στοίβας; Ποιους άλλους χώρους δεδομένων γνωρίζετε, τους οποίους χρησιμοποιεί ένα πρόγραμμα κατά τη λειτουργία του, και ποια είναι η χρησιμότητα του καθενός απ΄ αυτούς; Διατυπώστε την απάντησή σας *σε 5 το πολύ γραμμές*.

   (β΄) Στην πρότυπη βιβλιοθήκη της C περιλαμβάνεται και η συνάρτηση

   ```
   char *strchr(const char *s, int c)
   ```

   Η συνάρτηση αυτή επιστρέφει ένα δείκτη στην πρώτη εμφάνιση του χαρακτήρα που έχει ASCII κωδικό `c` μέσα στη συμβολοσειρά `s`. Αν δεν υπάρχει τέτοιος χαρακτήρας στην `s`, επιστρέφει `NULL`. Δώστε μία πιθανή υλοποίηση της συνάρτησης αυτής, **χωρίς να χρησιμοποιήσετε άλλες συναρτήσεις της πρότυπης βιβλιοθήκης**.

2. Για να χρησιμοποιήσουμε στην C μία λίστα ακεραίων, συνήθως ορίζουμε τον κόμβο της ως εξής:

   ```
   typedef struct listnode *Listptr;
   struct listnode {
      int value;
      Listptr next; }
   ```

   (α΄) Αν θέλουμε να ορίσουμε συναρτήσεις για την εύρεση αν ένας ακέραιος περιέχεται σε μία λίστα και για την εισαγωγή ακεραίου στην αρχή, αλλά και στο τέλος, μίας λίστας, τα πρωτότυπα των σχετικών συναρτήσεων θα ήταν πολύ εύλογο να ορισθούν ως εξής:

   ```
   int in(Listptr, int);
   void insert_at_start(Listptr *, int);
   void insert_at_end(Listptr *, int);
   ```

   Εξηγήστε, *σε 5 το πολύ γραμμές*, γιατί στις δύο τελευταίες περιπτώσεις έχουμε "`Listptr *`", ενώ στην πρώτη αρκεί το "`Listptr`".

   (β΄) Ποια χρήσιμη λειτουργία κάνει η συνάρτηση C που δίνεται στη συνέχεια; Αν την καλούσατε για μία λίστα τριών ακεραίων, δείξτε σχηματικά ποια θα ήταν η κατάσταση της λίστας αυτής στο τέλος κάθε επανάληψης του `while`, καθώς και στο τέλος της εκτέλεσης της συνάρτησης.

   ```
   void play_with_list(Listptr *ptraddr)
   { Listptr p, q;
     p = NULL;
     while (*ptraddr != NULL) {
       q = (*ptraddr)->next;
       (*ptraddr)->next = p;
       p = *ptraddr;
       *ptraddr = q; }
     *ptraddr = p; }
   ```

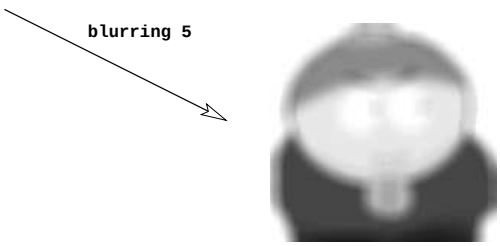
3. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`pythagorean.c`"), το οποίο να παίρνει από τη γραμμή εντολής δύο θετικούς ακεραίους m και n (με $m < n$) και να βρίσκει όλα τα *Πυθαγόρεια τρίγωνα* με μήκη πλευρών μεταξύ m και n, συμπεριλαμβανομένων. Υπενθυμίζεται ότι Πυθαγόρειο είναι ένα τρίγωνο όταν είναι ορθογώνιο και οι πλευρές του έχουν ακέραια μήκη. Εννοείται ότι κάθε τρίγωνο πρέπει να βρίσκεται μία μόνο φορά και όχι με όλες τις αναδιατάξεις των πλευρών του.

Το πρόγραμμά σας να βρίσκει επίσης ποιο από τα Πυθαγόρεια τρίγωνα που ανακάλυψε προσομοιάζει περισσότερο σε ισοσκελές, δηλαδή οι οξείες γωνίες του είναι πιο κοντά στις 45^0 από αυτές των άλλων τριγώνων, και ποιο είναι το πλέον ασύμμετρο, δηλαδή έχει τη μικρότερη οξεία γωνία από όλα τα άλλα Πυθαγόρεια τρίγωνα που βρέθηκαν. Αν υπάρχουν περισσότερα του ενός τρίγωνα που προσομοιάζουν εξ ίσου με ισοσκελές ή είναι εξ ίσου πλέον ασύμμετρα, δηλαδή είναι όμοια μεταξύ τους, αρκεί να βρίσκετε ένα μόνο απ΄ αυτά. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
% ./pythagorean 3 40
   3    4    5
   5   12   13
   6    8   10
   7   24   25
...............
  24   32   40
Most  symmetric triangle is:   20   21   29
Most asymmetric triangle is:    7   24   25
```

4. Υποθέστε ότι μία ασπρόμαυρη εικόνα, αλλά με αποχρώσεις του γκρι, μπορεί να αναπαρασταθεί με ένα διδιάστατο πίνακα, όπου η τιμή κάθε στοιχείου του, έστω από 0 (μαύρο) έως 255 (άσπρο), είναι το επίπεδο μαύρου-άσπρου του αντίστοιχου εικονοστοιχείου (pixel) της εικόνας. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`blurring.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (`stdin`) μία εικόνα, πρώτα τις διαστάσεις της (ύψος και πλάτος) και στη συνέχεια τον πίνακα των εικονοστοιχείων της, και να εκτυπώνει στην πρότυπη έξοδο (`stdout`) την εικόνα "θολωμένη". Υλοποιήστε τον απλό αλγόριθμο θόλωσης εικόνων που περιγράφεται στη συνέχεια. Για κάθε εικονοστοιχείο στην αρχική εικόνα, υπολογίζεται ο μέσος όρος των τιμών του ίδιου και όλων των εικονοστοιχείων που βρίσκονται μέσα στο νοητό τετράγωνο, στην αρχική εικόνα, με κέντρο το συγκεκριμένο εικονοστοιχείο και πλευρές που απέχουν απόσταση d από αυτό. Η τιμή αυτού του εικονοστοιχείου στη θολωμένη εικόνα είναι το ακέραιο μέρος αυτού του μέσου όρου. Το d δίνεται σαν όρισμα στη γραμμή εντολής. Παρακάτω, αριστερά είναι ένα παράδειγμα εκτέλεσης και δεξιά το αποτέλεσμα της θόλωσης μίας εικόνας (145×150) με αυτόν τον αλγόριθμο.

```
% cat image.txt
5 7
  0   0   0 190   0    0   0
 20  30  40 210 100   50  30
 40  60  80 230 150   90  60
  0   0   0 250   0    0   0
  0   0   0 255   0    0   0
% ./blurring 2 < image.txt
5 7
 30  75  76  82  82   92  53
 22  71  70  74  74   85  40
 18  70  66  69  69   80  32
 22  75  73  77  77   89  40
 20  76  71  74  74   86  33
```



Παρατηρήστε, στην εκτέλεση αριστερά, ότι $(0+0+0+20+30+40+40+60+80)/9 = 30$, που είναι η τιμή του εικονοστοιχείου στην επάνω αριστερά θέση του αποτελέσματος. Από το παράδειγμα αυτό βλέπετε ότι αν το νοητό τετράγωνο που έχει κέντρο το εικονοστοιχείο που μας ενδιαφέρει εκτείνεται έξω από τα όρια της εικόνας, προφανώς τα εικονοστοιχεία για τα οποία υπολογίζεται ο μέσος όρος της τιμής τους είναι λιγότερα από αυτά που θα είχαμε αν το εικονοστοιχείο μας ήταν πιο κεντρικό και το νοητό τετράγωνο βρισκόταν όλο εντός της εικόνας. Για παράδειγμα, το δεύτερο συμβαίνει για το 66 (τρίτη γραμμή και τρίτη στήλη του αποτελέσματος), που ισούται με το ακέραιο μέρος του 1/25ου του αθροίσματος των τιμών των πέντε πρώτων στηλών στον πίνακα της αρχικής εικόνας (απόσταση πλευρών από το 80 ίση με 2), αφού $(0+0+0+190+0+20+30+40+210+100+40+60+80+230+150+0+0+0+250+0+0+0+0+255+0)/25=66.2$.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2008 (17/9/2008)

1. (α΄) Ποιες είναι οι διαφορές, στη φύση και τη λειτουργία τους, των παρακάτω δύο γραμμών κώδικα σ΄ ένα πρόγραμμα C;

```
#define LIMIT 10

int limit = 10;
```

(β΄) Πιστεύετε ότι οι παρακάτω δύο γραμμές κώδικα σ΄ ένα πρόγραμμα C είναι ισοδύναμες, δηλαδή θα δώσουν το ίδιο αποτέλεσμα στην έξοδο, σε κάθε περίπτωση; (Το `i` είναι μία ακέραια μεταβλητή).

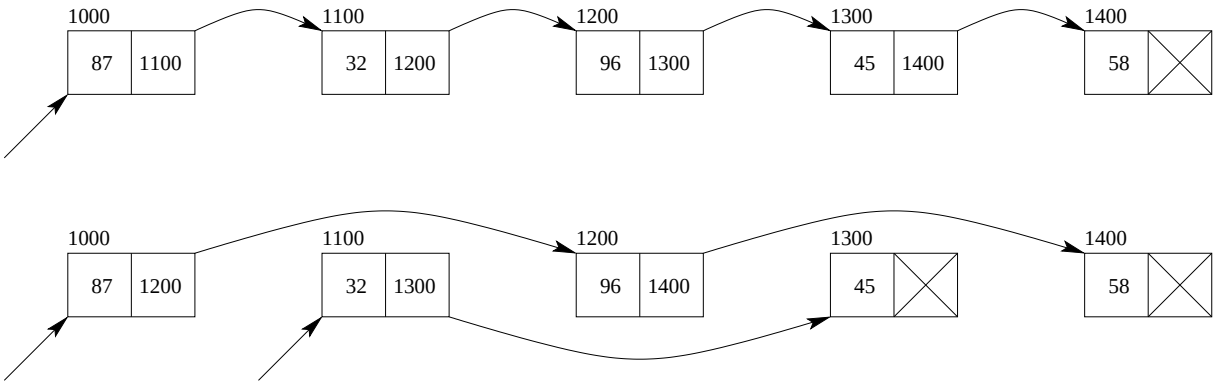
```
printf("%d\n%d\n", ++i, i++);

printf("%d\n", ++i); printf("%d\n", i++);
```

Δικαιολογήστε την απάντησή σας, δίνοντας και αποτελέσματα από τη εκτέλεση των γραμμών αυτών (φυσικά, θεωρώντας δεδομένη κάποια συγκεκριμένη τιμή για τη μεταβλητή `i`).

2. Για να χρησιμοποιήσουμε στην C μία λίστα ακεραίων, συνήθως ορίζουμε τον κόμβο της ως εξής:

```
typedef struct listnode *Listptr;

struct listnode {
   int value;
   Listptr next;
}
```

Ορίστε το πρωτότυπο και υλοποιήστε σε C μία συνάρτηση, η οποία θα διαχωρίζει τα στοιχεία μίας δεδομένης λίστας ακεραίων σε δύο λίστες, η μία να αποτελείται από τα περιττής τάξης στοιχεία της αρχικής (1ο, 3ο, 5ο, κλπ.) και η άλλη από τα άρτιας τάξης στοιχεία (2ο, 4ο, κλπ.). Η συνάρτησή σας **να μην δημιουργεί** νέες λίστες, δηλαδή **να μην αντιγράφει** τους κόμβους της αρχικής σε νέες θέσεις μνήμης. Στο παρακάτω σχήμα φαίνεται επάνω η αρχική λίστα και κάτω οι δύο νέες λίστες. Παρατηρήστε ότι οι κόμβοι δεν μετακινήθηκαν, απλώς έγιναν οι κατάλληλες διασυνδέσεις για να προκύψουν οι νέες λίστες. Να σημειωθεί ότι δεν μας ενδιαφέρει αν η αρχική λίστα θα υφίσταται πλέον, μετά την κλήση της συνάρτησης. Δεν χρειάζεται να υλοποιήσετε `main` συνάρτηση που να καλεί αυτήν που θα γράψετε.



Πώς θα ορίζατε το πρωτότυπο της συνάρτησης (δεν χρειάζεται να την υλοποιήσετε) αν έπρεπε η αρχική λίστα να παραμείνει ανέπαφη και οι νέες λίστες να αποτελούνται από νέους κόμβους; Στην περίπτωση αυτή, θα έπρεπε να γίνει, φυσικά, δυναμική δέσμευση μνήμης για τους νέους κόμβους. Επίσης, οι τιμές του μέλους `value` για τους κόμβους αυτούς θα προέκυπταν από αντιγραφή αυτών της αρχικής λίστας.

3. (α′) Υλοποιήστε σε C μία συνάρτηση `int prime(int)`, η οποία να ελέγχει αν ο ακέραιος που της δίνεται σαν όρισμα είναι πρώτος αριθμός και να επιστρέφει κατάλληλη τιμή στο όνομά της. Θα εκτιμηθεί όχι μόνο η ορθότητα της απάντησής σας, αλλά και η αποδοτικότητα της συνάρτησης που θα γράψετε.

   (β′) Δύο πρώτοι αριθμοί λέγονται δίδυμοι όταν διαφέρουν κατά 2 (π.χ. 3 και 5, 17 και 19, κλπ.). Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`twinpr.c`"), το οποίο να παίρνει από τη γραμμή εντολής ένα θετικό ακέραιο N και να βρίσκει και να εκτυπώνει τα πρώτα N ζεύγη διδύμων πρώτων αριθμών. Μία ενδεικτική εκτέλεση φαίνεται στη συνέχεια:

```
% ./twinpr 8170
3 5
5 7
11 13
17 19
29 31
.................
999611 999613
999959 999961
1000037 1000039
%
```

   Ερώτημα για έξτρα βαθμό 3%: Παραβλέποντας το θέμα των περιορισμών στην αναπαράσταση αριθμών στον υπολογιστή και θεωρώντας ότι έχετε απεριόριστο διαθέσιμο χρόνο για εκτέλεση του προγράμματος, είναι πιθανό αυτό να μην τερματίσει ποτέ, αν του δοθεί αρκετά μεγάλο N; Δικαιολογήστε την απάντησή σας.

4. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`helix.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (`stdin`) ένα θετικό ακέραιο N και να εκτυπώνει στην έξοδο μία "ελικοειδή" διάταξη των αριθμών από το 1 έως το N^2, σ′ ένα τετράγωνο πλαίσιο $N \times N$, με τον εξής τρόπο. Οι αριθμοί διατάσσονται σε αύξουσα σειρά, ξεκινώντας από την πρώτη γραμμή του πλαισίου από αριστερά προς τα δεξιά, μετά την τελευταία στήλη από επάνω προς τα κάτω, μετά την τελευταία γραμμή από δεξιά προς τα αριστερά, κ.ο.κ., μέχρι τον τελευταίο αριθμό N^2, που θα βρίσκεται κάπου στο κέντρο του πλαισίου. Η ενδεικτική εκτέλεση που ακολουθεί αποσαφηνίζει πλήρως το ζητούμενο.

```
% ./helix
Please, give size of helix: 15
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
  56  57  58  59  60  61  62  63  64  65  66  67  68  69  16
  55 104 105 106 107 108 109 110 111 112 113 114 115  70  17
  54 103 144 145 146 147 148 149 150 151 152 153 116  71  18
  53 102 143 176 177 178 179 180 181 182 183 154 117  72  19
  52 101 142 175 200 201 202 203 204 205 184 155 118  73  20
  51 100 141 174 199 216 217 218 219 206 185 156 119  74  21
  50  99 140 173 198 215 224 225 220 207 186 157 120  75  22
  49  98 139 172 197 214 223 222 221 208 187 158 121  76  23
  48  97 138 171 196 213 212 211 210 209 188 159 122  77  24
  47  96 137 170 195 194 193 192 191 190 189 160 123  78  25
  46  95 136 169 168 167 166 165 164 163 162 161 124  79  26
  45  94 135 134 133 132 131 130 129 128 127 126 125  80  27
  44  93  92  91  90  89  88  87  86  85  84  83  82  81  28
  43  42  41  40  39  38  37  36  35  34  33  32  31  30  29
%
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2009 (11/9/2009)

1. Απαντήστε στα παρακάτω ερωτήματα του συγκεκριμένου θέματος σε 20 το πολύ γραμμές συνολικά.

   (α΄) Ποια είναι η σχέση δεικτών (pointers) και στατικών πινάκων στη γλώσσα προγραμματισμού C;

   (β΄) Θα μπορούσαμε σε ένα αυθαίρετο πρόγραμμα C, στο οποίο χρησιμοποιούνται και δείκτες και στατικοί πίνακες, να αντικαταστήσουμε τη χρήση των πινάκων με δείκτες; Αν ναι, πώς (για μονοδιάστατους πίνακες); Αν όχι, γιατί; Θα μπορούσε να γίνει και το αντίστροφο, δηλαδή αντικατάσταση της χρήσης δεικτών με στατικούς πίνακες; Αν ναι, πώς; Αν όχι, γιατί;

   (γ΄) Έστω ότι κάποιος ισχυρίζεται πως παρότι είναι γενικά ωφέλιμο να χρησιμοποιούμε συναρτήσεις στον προγραμματισμό σε C, το να γράψουμε μία συνάρτηση που θα επεξεργάζεται έναν πίνακα δεν ενδείκνυται, γιατί, αν ο πίνακας είναι πολύ μεγάλος, μπορεί να είναι πρακτικά ασύμφορο να περάσουμε τα στοιχεία του στη συνάρτηση. Σχολιάστε αυτή την άποψη.

   (δ΄) Έστω το παρακάτω τμήμα προγράμματος:

```
int i, *p, x[] = {8, 2, 3, 5, 9};
for (i = 1 ; i <= 5 ; i++) {
   *p = x[i];
   printf("%d\n", *p); }
```

   Υπάρχουν συντακτικά λάθη στο παραπάνω τμήμα προγράμματος; Αν ναι, διορθώστε τα. Αφού τα διορθώσετε, ή αν δεν υπάρχουν καν, ποια πιστεύετε ότι είναι η πιο πιθανή συμπεριφορά του εκτελέσιμου προγράμματος που θα προέκυπτε από το παραπάνω; Θα παίρναμε κάποια συγκεκριμένα αποτελέσματα, ή όχι; Σε κάθε περίπτωση, εξηγήστε την απάντησή σας.

2. (α΄) Γνωρίζετε ότι μπορούμε στον προγραμματισμό σε C να έχουμε λίστες από ακεραίους, αλλά και δυαδικά δέντρα από ακεραίους. Ας υποθέσουμε ότι σε κάποιες εφαρμογές θα ήταν χρήσιμο να ορίσουμε και λίστες από δυαδικά δέντρα ακεραίων ή και δυαδικά δέντρα από λίστες ακεραίων. Σχεδιάστε δύο σχήματα που να δείχνουν πώς φαντάζεστε κάποιες τέτοιες δομές δεδομένων και δώστε τους απαιτούμενους ορισμούς `struct` σε C που θα χρειαζόταν κάποιος για να διαχειριστεί αυτές τις δομές.

   (β΄) Συνηθίζουμε στα **ταξινομημένα** δυαδικά δέντρα να μην έχουμε, για λόγος καλύτερης διαχείρισής τους, κόμβους που επαναλαμβάνονται παραπάνω από μία φορά στο δέντρο. Αν, παρ΄ όλα αυτά, σε κάποια εφαρμογή χρειαζόμαστε ένα ταξινομημένο δυαδικό δέντρο, στο οποίο απαιτείται να εισάγουμε ένα κόμβο πολλές φορές, και η πληροφορία της πολλαπλής εισαγωγής να φυλάσσεται, έχουμε την εξής δυνατότητα: Μπορούμε σε κάθε κόμβο του δέντρου να κρατάμε και την πληροφορία πόσες φορές έχει εισαχθεί ο κόμβος, αντί να έχουμε πολλαπλές εμφανίσεις του μέσα στο δέντρο.

   Δώστε τον απαιτούμενο ορισμό `struct` για τους κόμβους ενός τέτοιου δυαδικού δέντρου ακεραίων και υλοποιήστε σε C μία συνάρτηση (με πρωτότυπο που θα αποφασίσετε εσείς) για την εισαγωγή ενός ακεραίου σε ένα τέτοιο ταξινομημένο δυαδικό δέντρο ακεραίων.

3. Ένας ακέραιος μεγαλύτερος του 1 ονομάζεται *τέλειος αριθμός* (perfect number) αν ισούται με το άθροισμα των διαιρετών του, συμπεριλαμβανομένης της μονάδας, αλλά εξαιρουμένου του ίδιου του αριθμού. Ο μικρότερος τέλειος αριθμός είναι το 6, αφού ισούται με το άθροισμα των διαιρετών του $(1 + 2 + 3)$. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`perfect.c`"), το οποίο να καλείται με έναν ακέραιο N σαν όρισμα στη γραμμή εντολής και να εκτυπώνει όλους τους τέλειους αριθμούς που είναι μικρότεροι ή ίσοι του N, καθώς επίσης και το πώς γράφονται σαν άθροισμα των διαιρετών τους. Ένα παράδειγμα εκτέλεσης φαίνεται στην επόμενη σελίδα.

```
% ./perfect 10000
6 = 1 + 2 + 3
28 = 1 + 2 + 4 + 7 + 14
496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248
8128 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 127 + 254 + 508 + 1016 + 2032 + 4064
```

Υπόδειξη/σημείωση: Το πρόβλημα μπορεί να λυθεί χωρίς τη χρήση πίνακα. Δεν απαγορεύεται η χρήση πινάκων στην απάντησή σας, αλλά σημειώστε ότι στην περίπτωση αυτή θα κάνετε άσκοπη σπατάλη μνήμης και αυτό θα αξιολογηθεί αναλόγως.

4. Σε κάποιο σύλλογο, π.χ. φοιτητικό, συνδικαλιστικό, κλπ., πρόκειται να γίνουν εκλογές για την κάλυψη των εδρών του Διοικητικού Συμβουλίου (ΔΣ) του συλλόγου. Στις εκλογές κατεβαίνουν παρατάξεις και το εκλογικό σύστημα που εφαρμόζεται είναι η απλή αναλογική. Αυτό σημαίνει ότι δεδομένων των ψήφων που πήρε η κάθε παράταξη στις εκλογές, οι έδρες κατανέμονται ως εξής:[1]

   Αρχικά υπολογίζεται το εκλογικό μέτρο, το οποίο ισούται με το πηλίκο της διαίρεσης του συνολικού αριθμού των ψήφων προς το πλήθος των εδρών στο ΔΣ. Στην πρώτη κατανομή, κάθε παράταξη παίρνει τόσες έδρες όσες το πηλίκο του αριθμού των ψήφων που έλαβε προς το εκλογικό μέτρο. Προφανώς, με την πρώτη κατανομή, δεν διατίθενται όλες οι έδρες και αρκετές παρατάξεις έχουν "*αδιάθετο υπόλοιπο ψήφων*". Το αδιάθετο υπόλοιπο ψήφων κάθε παράταξης ισούται με το υπόλοιπο της διαίρεσης του αριθμού των ψήφων που έλαβε προς το εκλογικό μέτρο. Στη δεύτερη κατανομή διατίθενται και οι έδρες που έμειναν αδιάθετες από την πρώτη, ανά μία σε κάθε παράταξη με φθίνουσα σειρά των αδιάθετων υπολοίπων ψήφων, μέχρι να διατεθούν όλες οι έδρες.

   Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`elections.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (stdin) τα αποτελέσματα των εκλογών σε ένα σύλλογο και να υπολογίζει τις έδρες που κερδίζει κάθε παράταξη. Στην πρώτη γραμμή της εισόδου θα δίνονται δύο ακέραιοι, που θα είναι το πλήθος των παρατάξεων που συμμετέχουν στις εκλογές και το πλήθος των εδρών του ΔΣ. Στη συνέχεια, θα δίνονται οι αριθμοί των ψήφων που έλαβε κάθε παράταξη, ένας αριθμός σε κάθε γραμμή (ή όλοι στην ίδια). Το πρόγραμμά σας να εκτυπώνει τελικά έναν πίνακα με τον αριθμό των ψήφων κάθε παράταξης, το ποσοστό της (με δύο δεκαδικά ψηφία), το πλήθος των εδρών που έλαβε (από την πρώτη και από την δεύτερη κατανομή) και το τυχόν τελικό πλήθος αδιάθετων ψήφων που της έμειναν.[2] Κάποιο παράδειγμα εκτέλεσης:

```
% ./elections
7 9
100 150 50 85 167 19 33

Total number of votes: 604
Quota for a seat:      67

Party     Votes   Percentage   Seats   Remainder
Party 1:   100      16.56%    2 (1+1)      0
Party 2:   150      24.83%    2 (2+0)     16
Party 3:    50       8.28%    1 (0+1)      0
Party 4:    85      14.07%    1 (1+0)     18
Party 5:   167      27.65%    3 (2+1)      0
Party 6:    19       3.15%    0 (0+0)     19
Party 7:    33       5.46%    0 (0+0)     33
```

[1] Υποθέστε ότι, για λόγους απλοποίησης της κατάστασης, δεν υπάρχουν άκυρα/λευκά ψηφοδέλτια στο αποτέλεσμα, ή, αν υπάρχουν, τα αγνοούμε για το πρόβλημα που συζητάμε.

[2] Σημειώστε ότι στην όχι απίθανη περίπτωση που πρέπει να διατεθεί κάποια έδρα στη δεύτερη κατανομή και υπάρχουν περισσότερες από μία παράταξη με το ίδιο μέγιστο αδιάθετο υπόλοιπο ψήφων, ισχύει ο εξής "ιδιόρρυθμος" κανόνας. Την έδρα την καταλαμβάνει η "αρχαιότερη" παράταξη. Οι ψήφοι των παρατάξεων δίνονται στην είσοδο κατά σειρά από την αρχαιότερη προς τη νεώτερη.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β′ Περιόδου 2010 (9/9/2010)

1. (α′) Γιατί στις απλές υλοποιήσεις του υπολογισμού του παραγοντικού ενός ακεραίου N, στη γλώσσα προγραμματισμού C, όπως αυτές που υπάρχουν στις σημειώσεις του μαθήματος, τα αποτελέσματα είναι λάθος για κάποια τιμή του N και τις μεγαλύτερες από αυτήν; Πιστεύετε ότι υπάρχει δυνατότητα να υλοποιηθεί σωστά ο υπολογισμός του $N!$ για οσοδήποτε μεγάλες τιμές του N;

   (β′) Τι θα εκτυπωθεί από το παρακάτω τμήμα προγράμματος C και γιατί;

```
int x = 5;
if (x = 0)
   printf("%d is zero\n", x);
else
   printf("%d is not zero\n", x);
```

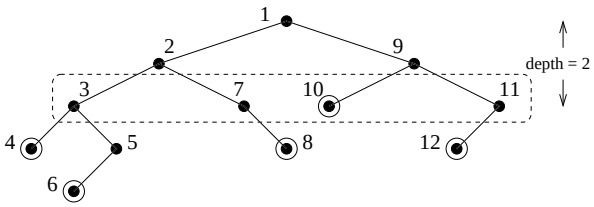
   (γ′) Πόση βεβαιότητα μπορεί να έχει κάποιος για το τι θα εκτυπωθεί από το παρακάτω τμήμα προγράμματος C; Εξηγήστε την άποψή σας.

```
int count = 0;
double x;
for (x = 0.0 ; x < 100000000.0 ; x++)
   if (sqrt(x) * sqrt(x) == x)
      count++;
printf("%d\n", count);
```

   (δ′) Γνωρίζετε ότι για να ανταλλαγούν οι τιμές δύο μεταβλητών χρειάζεται μία βοηθητική μεταβλητή. Παρόλα αυτά, μπορείτε να δώσετε μία αλληλουχία εντολών C, με την οποία να ανταλλάσσονται οι τιμές των ακέραιων μεταβλητών `a` και `b`, **χωρίς τη χρήση βοηθητικής μεταβλητής**;

2. Στην C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου ακεραίων ως εξής:

```
typedef struct treenode *Treeptr;

struct treenode {
  int value;
  Treeptr left;
  Treeptr right;
};
```



   (α′) Στο παραπάνω σχήμα, οι αριθμοί που φαίνονται στους κόμβους του δυαδικού δέντρου υποδεικνύουν τη σειρά επίσκεψης των κόμβων κατά τη λεγόμενη *προδιατεταγμένη διάσχιση* (preorder traversal). Προσπαθήστε να κατανοήσετε τη λογική της διάσχισης αυτής και υλοποιήστε μία συνάρτηση C, με πρωτότυπο `void print_preorder(Treeptr)`, η οποία να παίρνει σαν όρισμα ένα δυαδικό δέντρο ακεραίων και να εκτυπώνει τα περιεχόμενα των κόμβων του σύμφωνα με τη προδιατεταγμένη διάσχιση. Για παράδειγμα, αν οι αριθμοί που φαίνονται στο παραπάνω δέντρο ήταν και τα αντίστοιχα περιεχόμενα των κόμβων, η συνάρτηση θα έπρεπε να εκτυπώσει: `1 2 3 4 5 6 7 8 9 10 11 12`.

   (β′) Υλοποιήστε μία συνάρτηση C, με πρωτότυπο `int count_nodes(Treeptr)`, η οποία να παίρνει σαν όρισμα ένα δυαδικό δέντρο ακεραίων και να επιστρέφει το πλήθος των κόμβων του δέντρου. Για παράδειγμα, για το παραπάνω δέντρο, η συνάρτηση πρέπει να επιστρέψει 12.

   (γ′) Υλοποιήστε μία συνάρτηση C, με πρωτότυπο `int count_leaves(Treeptr)`, η οποία να παίρνει σαν όρισμα ένα δυαδικό δέντρο ακεραίων και να επιστρέφει το πλήθος των **κόμβων-φύλλων** του δέντρου, δηλαδή αυτών που δεν έχουν ούτε αριστερό, ούτε δεξί παιδί. Για παράδειγμα, για το παραπάνω δέντρο, η συνάρτηση πρέπει να επιστρέψει 5 (οι κόμβοι-φύλλα είναι οι 4, 6, 8, 10 και 12).

(δ′) Υλοποιήστε μία συνάρτηση C, με πρωτότυπο `void print_level(Treeptr, int)`, η οποία να παίρνει σαν όρισμα ένα δυαδικό δέντρο ακεραίων και έναν ακέραιο (έστω `depth`) και να εκτυπώνει τα περιεχόμενα των κόμβων του δέντρου που βρίσκονται στο επίπεδο βάθους `depth`. Θεωρήστε ότι η ρίζα του δέντρου είναι στο επίπεδο βάθους 0, τα παιδιά της στο επίπεδο βάθους 1, κ.ο.κ. Για το παραπάνω δέντρο, αν κληθεί η συνάρτηση με `depth = 2`, θα πρέπει να εκτυπώσει: `3 7 10 11`.

3. Ένας ακέραιος μεγαλύτερος του 1 λέγεται ότι είναι "άσχημος" όταν έχει σαν μόνους πρώτους διαιρέτες το 2, το 3 και το 5 (κάποιους από αυτούς ή όλους). Οι αριθμοί που δεν είναι "άσχημοι" είναι "όμορφοι". Για παράδειγμα, το 20 είναι "ασχημος" αριθμός, αφού $20 = 2 \times 2 \times 5$, όπως και το 90, αφού $90 = 2 \times 3 \times 3 \times 5$. Όμως, το 42 είναι "όμορφος" αριθμός, αφού $42 = 2 \times 3 \times 7$ (δηλαδή έχει σαν διαιρέτη και το 7). Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`ugly.c`"), το οποίο να καλείται με έναν ακέραιο N στη γραμμή εντολής, μεγαλύτερο του 1, και να εκτυπώνει όλους τους "άσχημους" αριθμούς από το 2 έως και το N. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
% ./ugly 70
2 3 4 5 6 8 9 10 12 15 16 18 20 24 25 27 30 32 36 40 45 48 50 54 60 64
```

4. Γνωρίζετε από τη Γραμμική Άλγεβρα ότι το γινόμενο δύο πινάκων $A = (a_{ik})$ και $B = (b_{kj})$ ορίζεται όταν οι πίνακες αυτοί έχουν διαστάσεις $N \times L$ και $L \times M$, αντίστοιχα. Δηλαδή, όταν το πλήθος των στηλών του πρώτου ισούται με το πλήθος των γραμμών του δεύτερου. Τότε, ο πίνακας-γινόμενο $C = (c_{ij})$ θα έχει διαστάσεις $N \times M$, ενώ ο υπολογισμός των στοιχείων του δίνεται από τον τύπο:

$$c_{ij} = \sum_{k=1}^{L} a_{ik} \cdot b_{kj}, \quad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, M$$

Δηλαδή, το στοιχείο που βρίσκεται στην i γραμμή και j στήλη του γινομένου ισούται με το άθροισμα των γινομένων των στοιχείων, ένα προς ένα, της i γραμμής του πρώτου πίνακα και της j στήλης του δεύτερου πίνακα.

Γράψτε μία συνάρτηση C (με πρωτότυπο δικής σας επιλογής), η οποία να υπολογίζει το γινόμενο δεδομένων πινάκων. Στη συνέχεια, γράψτε και κυρίως πρόγραμμα, το οποίο να διαβάζει από την πρότυπη είσοδο (stdin) τρεις θετικούς ακεραίους N, L και M και στη συνέχεια να δημιουργεί δύο πίνακες ακεραίων A και B διαστάσεων $N \times L$ και $L \times M$, αντίστοιχα, με τυχαία στοιχεία στο διάστημα $[0, 99]$. Αφού εκτυπώσει τους πίνακες αυτούς, να καλεί την προηγούμενη συνάρτηση για τον υπολογισμό του γινομένου τους, το οποίο και να εκτυπώνει. Ένα παράδειγμα εκτέλεσης (έστω ότι το πηγαίο αρχείο σας είναι το "`matrmult.c`") δίνεται στη συνέχεια:

```
% ./matrmult
Please, give dimensions: 4 2 3
Matrix A is:
 38 58
 13 15
 51 27
 10 19
Matrix B is:
 12 86 49
 67 84 60
Matrix C (product of A and B) is:
 4342 8140 5342
 1161 2378 1537
 2421 6654 4119
 1393 2456 1630
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2011 (15/10/2011)

1. (α΄) Σχολιάστε την επόμενη πρόταση: "*Στη γλώσσα C, οι εξωτερικές/καθολικές μεταβλητές είναι πολύ χρήσιμες, διότι μέσω αυτών μπορούμε να περιορίσουμε τα ορίσματα στις συναρτήσεις για την ανταλλαγή δεδομένων μεταξύ τους, οπότε τα προγράμματά μας εκτελούνται πολύ ταχύτερα.*"

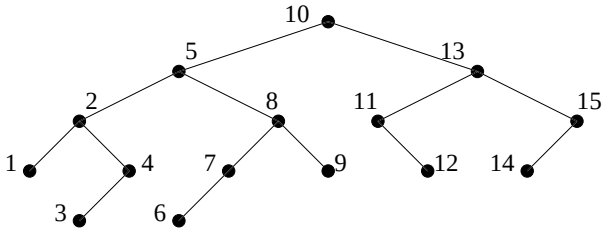
   (β΄) Ποια είναι η διαφορά των εντολών `break` και `continue` όταν αυτές χρησιμοποιούνται μέσα σε βρόχους `for`, `while` ή `do/while`; Δείξτε μέσω ενός παραδείγμα-τος/-των κώδικα C τη διαφορά των δύο εντολών, δίνοντας το τελικό αποτέλεσμα της εκτέλεσης του κώδικα που δώσατε.

   (γ΄) Έστω ότι θέλουμε σε ένα πρόγραμμα C να σχολιάσουμε προσωρινά ένα μεγάλο τμήμα του, το οποίο περιέχει και σχόλια (πιθανώς για λόγους αποσφαλμάτωσης). Έχουμε κάποιο τρόπο να το επιτύχουμε αυτό, δεδομένου ότι στην C δεν επιτρέπονται εμφωλευμένα σχόλια μέσα σε σχόλια;

   (δ΄) Ορίστε σε C τη συνάρτηση `int bigmod(int b, int p, int m)`, η οποία να υπολογίζει, αν είναι δυνατόν με τον πλέον αποδοτικό τρόπο, και να επιστρέφει το $b^p \ mod \ m$, όπου mod είναι το υπόλοιπο διαίρεσης. Σημειώστε ότι οι ακέραιοι b και p μπορεί να είναι ιδιαίτερα μεγάλοι, οπότε το b^p ενδέχεται να μην είναι δυνατόν να καταχωρηθεί στον υπολογιστή σε χώρο ενός ακεραίου. **Υπόδειξη**: $(x \cdot y) \ mod \ z = ((x \ mod \ z) \cdot (y \ mod \ z)) \ mod \ z$

2. Στην C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου ακεραίων ως εξής:

```
typedef struct treenode *Treeptr;

struct treenode {
  int value;
  Treeptr left;
  Treeptr right;
};
```



Σχήμα 1

   (α΄) Στο Σχήμα 1, οι αριθμοί που φαίνονται στους κόμβους του δυαδικού δέντρου υποδεικνύουν τη σειρά επίσκεψης των κόμβων κατά τη λεγόμενη *ενδοδιατεταγμένη διάσχιση* (inorder traversal). Προσπαθήστε να κατανοήσετε τη λογική της διάσχισης αυτής και υλοποιήστε μία συνάρτηση C, με πρωτότυπο `void print_inorder(Treeptr)`, η οποία να παίρνει σαν όρισμα ένα δυαδικό δέντρο ακεραίων και να εκτυπώνει τα περιεχόμενα των κόμβων του σύμφωνα με τη ενδοδιατεταγμένη διάσχιση. Για παράδειγμα, αν οι αριθμοί που φαίνονται στο παραπάνω δέντρο ήταν και τα αντίστοιχα περιεχόμενα των κόμβων, η συνάρτηση θα έπρεπε να εκτυπώσει:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

   (β΄) Υλοποιήστε μία συνάρτηση C, με πρωτότυπο `void print_paths(Treeptr)`, η οποία σε δεδομένο δυαδικό δέντρο ακεραίων που παίρνει σαν όρισμα, για κάθε μονοπάτι από τη ρίζα του δέντρου σε φύλλο του, να εκτυπώνει τα περιεχόμενα των κόμβων του μονοπατιού, ένα μονοπάτι σε κάθε γραμμή. Αν η συνάρτηση καλείτο για το δέντρο του Σχήματος 1, θα έπρεπε να εκτυπώσει:

```
10 5 2 1
10 5 2 4 3
10 5 8 7 6
10 5 8 9
10 13 11 12
10 13 15 14
```

   Μπορείτε να υποθέσετε, για ευκολία σας, ότι το δέντρο έχει ένα μέγιστο βάθος, π.χ. 1000.

3. Ένας θετικός ακέραιος ονομάζεται *ισχυρά σύνθετος* (highly composite) αν έχει περισσότερους διαιρέτες από κάθε άλλο θετικό ακέραιο μικρότερο από αυτόν. Για παράδειγμα, το 12 είναι ισχυρά σύνθετος αριθμός, αφού έχει 6 διαιρέτες (τους 1, 2, 3, 4, 6 και 12), και οποιοσδήποτε θετικός ακέραιος μικρότερος από το 12 έχει λιγότερους από 6 διαιρέτες. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`hcn.c`"), το οποίο να καλείται με έναν ακέραιο N σαν όρισμα στη γραμμή εντολής και να εκτυπώνει όλους του θετικούς ακεραίους που είναι ισχυρά σύνθετοι και μικρότεροι ή ίσοι του N. Μία ενδεικτική εκτέλεση είναι η εξής:

```
% ./hcn 11000
1 2 4 6 12 24 36 48 60 120 180 240 360 720 840 1260 1680 2520 5040 7560 10080
%
```

Υπόδειξη/σημείωση: Το πρόβλημα μπορεί να λυθεί χωρίς τη χρήση πίνακα. Δεν απαγορεύεται η χρήση πινάκων στην απάντησή σας, αλλά σημειώστε ότι στην περίπτωση αυτή θα κάνετε άσκοπη σπατάλη μνήμης και αυτό θα αξιολογηθεί αναλόγως.

4. Στο *Παιγνίδι της Ζωής* (Game of Life), που προτάθηκε από τον Άγγλο μαθηματικό John Horton Conway το 1970, έχουμε ένα ορθογώνιο δισδιάστατο πλαίσιο από κύτταρα, καθένα από τα οποία μπορεί να είναι είτε ζωντανό είτε νεκρό. Το παιγνίδι ξεκινά με μία αρχική κατάσταση των κυττάρων και εξελίσσεται αλλάζοντας κατάσταση σε κάποια από τα κύτταρα, ανάλογα με την κατάσταση των γειτονικών κυττάρων του καθενός, σύμφωνα με τους κανόνες που θα περιγραφούν στη συνέχεια. Ως γειτονικά κύτταρα ενός κυττάρου θεωρούνται όλα τα αμέσως διπλανά του, οριζόντια, κατακόρυφα ή διαγώνια. Οι κανόνες, που εφαρμόζονται ταυτόχρονα σε όλα τα κύτταρα, για να μεταβεί το παιγνίδι από μία κατάσταση στην επόμενή του, είναι οι εξής:

   - Κάθε ζωντανό κύτταρο με λιγότερους από δύο ή περισσότερους από τρεις ζωντανούς γείτονες πεθαίνει (λόγω υπο- και υπερ-πληθυσμού, αντίστοιχα).
   - Κάθε νεκρό κύτταρο με ακριβώς τρεις ζωντανούς γείτονες ζωντανεύει (λόγω αναπαραγωγής).
   - Όλα τα υπόλοιπα κύτταρα (ζωντανά ή νεκρά) παραμένουν ως έχουν και στην επόμενη κατάσταση.

   Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`gol.c`"), το οποίο να δέχεται στη γραμμή εντολής τρεις ακεραίους N, M και G και να δείχνει την εξέλιξη του Παιχνιδιού της Ζωής, σε ένα πλαίσιο με N γραμμές και M στήλες, για G καταστάσεις μετά την αρχική. Η αρχική κατάσταση του παιγνιδιού να δίνεται από την πρότυπη είσοδο (stdin), σαν ένας δισδιάστατος πίνακας $N \times M$ από `X` και `.` (τελείες), τα οποία παριστάνουν ένα ζωντανό και ένα νεκρό κύτταρο, αντίστοιχα. Μία ενδεικτική εκτέλεση φαίνεται στη συνέχεια. Μπορείτε να επιλέξετε να κάνετε τις εκτυπώσεις με κατακόρυφο τρόπο (κάθε κατάσταση κάτω από την προηγούμενή της) και όχι οριζόντια, όπως φαίνεται στην ενδεικτική εκτέλεση. Πώς επηρεάζει την υλοποίηση του προγράμματός σας το ποια μορφή εκτύπωσης θα επιλέξετε;

```
% ./gol 5 6 4
Please, give initial state
......
..XX..
...X..
.XXX..
......

State 0  --> State 1  --> State 2  --> State 3  --> State 4
......       ......       ......       ......       ......
..XX..       ..XX..       ..XX..       ..XX..       ..XX..
...X..       .X..X.       .X..X.       .X..X.       .X..X.
.XXX..       ..XX..       .XXX..       .X..X.       XX.XX.
......       ..X...       ..XX..       .X.X..       ..X...
%
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β′ Περιόδου 2012 (3/9/2012)

1. (α′) i. Ποια είναι η διαφορά των δομών επανάληψης `while` και `do...while` στη γλώσσα C; Με βάση ποια κριτήρια επιλέγουμε να χρησιμοποιήσουμε τη μία ή την άλλη;

   ii. Πώς μπορούμε να υλοποιήσουμε στη C έναν ατέρμονα βρόχο με την εντολή `for`; Πώς με την εντολή `while`; Ένα πρόγραμμα που περιέχει ατέρμονα βρόχο είναι καταδικασμένο να μην τερματίσει ποτέ; Δικαιολογήστε την απάντησή σας.

   (β′) Χωρίς να αλλάξετε το πρωτότυπο της διπλανής συνάρτησης C, ξαναγράψτε τον κώδικα στο σώμα της, χρησιμοποιώντας το πολύ 15 χαρακτήρες, έτσι ώστε να διατηρήσει την ίδια ακριβώς λειτουργικότητα. Δικαιολογήστε την απάντησή σας.

   (γ′) Ορίστε σε C μία συνάρτηση με πρωτότυπο

   `char *strrand(char *str)`

   η οποία να δέχεται σαν όρισμα μία συμβολοσειρά `str` και να την επιστρέφει (και στο όνομά της) με το περιεχόμενό της "επαρκώς" ανακατεμένο. Η επιλογή κατάλληλου αλγορίθμου αφήνεται στην κρίση σας. Εννοείται ότι τελικά η συμβολοσειρά θα περιέχει ακριβώς τους ίδιους χαρακτήρες που είχε αρχικά, απλώς "επαρκώς" ανακατεμένους.

```
int mystery(int n)
{
   int s;
   n = (n >= 0) ? (n + 9) : -n;
   while (n >= 10) {
      s = 0;
      while (n) {
         s += n % 10;
         n = n / 10;
      }
      n = s;
   }
   return (n == 9);
}
```

2. Εκτός από τις λίστες που γνωρίζετε ήδη, μπορούμε στον προγραμματισμό να χρησιμοποιήσουμε και κυκλικές λίστες. Οι κυκλικές λίστες διαφέρουν από τις απλές ως προς το ότι ο δείκτης του τελευταίου κόμβου δείχνει στον αρχικό, αντί να είναι `NULL`.

   (α′) Ορίστε τη δομή ενός κόμβου κυκλικής λίστας ακεραίων, αν υπάρχει λόγος να είναι διαφορετική από τη δομή ενός κόμβου απλής λίστας ακεραίων.

   (β′) Υλοποιήστε συνάρτηση C, με πρωτότυπο της επιλογής σας, η οποία να μετατρέπει μία απλή λίστα ακεραίων σε κυκλική.

   (γ′) Υλοποιήστε συνάρτηση C, με πρωτότυπο της επιλογής σας, η οποία να εισάγει σε μία κυκλική λίστα έναν κόμβο με τιμή Val αμέσως μετά το N-οστό στοιχείο της.[1]

   (δ′) Δώστε μόνο το πρωτότυπο συνάρτησης C που να διαγράφει το N-οστό στοιχείο (με την έννοια του προηγούμενου ερωτήματος) μίας κυκλικής λίστας ακεραίων.

   (ε′) Αναφέρατε κάποια περίπτωση εφαρμογής που θα ήταν προτιμότερο να χρησιμοποιηθεί κυκλική λίστα αντί μη κυκλικής.

3. Οι αριθμοί *αρμονικών διαιρετών* (ή αριθμοί Ore, προς τιμήν του Νορβηγού μαθηματικού, που τους όρισε το 1948) είναι εκείνοι οι θετικοί ακέραιοι για τους οποίους ισχύει ότι ο μέσος αρμονικός των διαιρετών τους (συμπεριλαμβανομένων του 1 και του εαυτού τους) είναι ακέραιος αριθμός. Δηλαδή, ένας αριθμός είναι Ore όταν έχει k διαιρέτες, τους d_1, d_2, …, d_k, και ισχύει ότι το κλάσμα

$$\frac{k}{\frac{1}{d_1}+\frac{1}{d_2}+\ldots+\frac{1}{d_k}}$$

   είναι ακέραιος αριθμός. Για παράδειγμα, το 6, που έχει διαιρέτες τους 1, 2, 3 και 6, είναι αριθμός Ore, αφού $\frac{4}{\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{6}} = 2$. Ομοίως και το 140, αφού $\frac{12}{\frac{1}{1}+\frac{1}{2}+\frac{1}{4}+\frac{1}{5}+\frac{1}{7}+\frac{1}{10}+\frac{1}{14}+\frac{1}{20}+\frac{1}{28}+\frac{1}{35}+\frac{1}{70}+\frac{1}{140}} = 5$. Γράψτε

[1] Η λίστα μπορεί να έχει και λιγότερα από N στοιχεία, οπότε η έννοια του N-οστού στοιχείου είναι συνυφασμένη με μία ή περισσότερες επαναδιασχίσεις της λίστας από την αρχή της.

ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`ore.c`"), το οποίο να διαβάζει από την πρότυπη είσοδο (stdin) ένα θετικό ακέραιο N και να εκτυπώνει όλους τους αριθμούς Ore που είναι μικρότεροι ή ίσοι του N. Μία ενδεικτική εκτέλεση του προγράμματος είναι η εξής:

```
% ./ore
Please, give maximum number: 50000
1 6 28 140 270 496 672 1638 2970 6200 8128 8190 18600 18620 27846 30240 32760
%
```

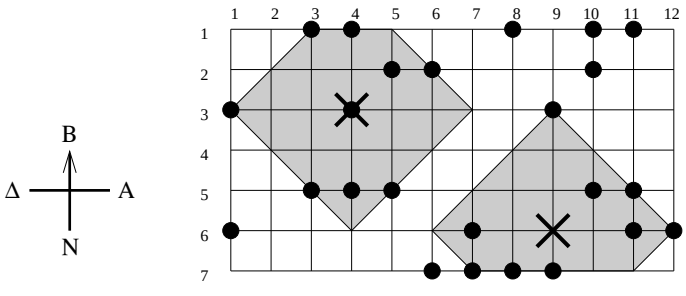
Σημείωση: Υπενθυμίζεται ότι όλοι σχεδόν οι πραγματικοί αριθμοί δεν μπορούν να αναπαρασταθούν με απόλυτη ακρίβεια στον υπολογιστή.

4. Η ρυμοτομία μίας πόλης είναι ένα ορθογώνιο πλέγμα διαστάσεων $N \times M$. Δηλαδή, η πόλη έχει N δρόμους στην κατεύθυνση Ανατολή-Δύση και M δρόμους στην κατεύθυνση Βορράς-Νότος. Οι αποστάσεις κάθε δρόμου από τους γειτονικούς παραλλήλους του είναι ίδιες, όση είναι και η πλευρά κάθε οικοδομικού τετραγώνου της πόλης. Οι κάτοικοι της πόλης αυτής είναι φανατικοί του καφέ, αλλά δεν τους αρέσει να περπατούν πολύ. Επίσης, θέλουν να έχουν και μεγάλη ποικιλία στις επιλογές τους για το πού να πιουν τον καφέ τους, όμως δεν είναι διατεθειμένοι να περπατήσουν περισσότερο από d τετράγωνα από το σπίτι τους, προφανώς κατά μήκος δρόμων της πόλης, μέχρι μία καφετέρια. Στην πόλη έχει ανοίξει ένας αριθμός από καφετέριες, σε διασταυρώσεις δρόμων, αλλά σε κάθε διασταύρωση υπάρχει το πολύ μία καφετέρια. Από τα προηγούμενα, είναι φανερό ότι τα σπίτια σε διασταυρώσεις που έχουν πολλές καφετέριες κοντά τους (σε απόσταση το πολύ d τετράγωνα) είναι αυτά που έχουν τη μεγαλύτερη ζήτηση για να μένει κάποιος.[2] Το ζητούμενο στο πρόβλημα αυτό είναι να βρεθεί, για δεδομένα N, M και d και δεδομένες τις θέσεις των καφετεριών στην πόλη, ποια είναι η διασταύρωση με τις περισσότερες καφετέριες σε απόσταση το πολύ d τετράγωνα.

   Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`coffee.c`"), το οποίο αφού δεχθεί από τη γραμμή εντολής τρεις ακεραίους N, M και d, να διαβάζει από την πρότυπη είσοδο (stdin) ένα διδιάστατο πίνακα χαρακτήρων $N \times M$, ο οποίος να παρέχει την πληροφορία για τις τοποθεσίες των καφετεριών στην πόλη. Όπου στον πίνακα αυτό δίνεται ο χαρακτήρας `'c'`, να θεωρείται ότι στην αντίστοιχη διασταύρωση υπάρχει καφετέρια. Οποιοσδήποτε άλλος χαρακτήρας σημαίνει έλλειψη καφετέριας στη διασταύρωση. Το πρόγραμμά σας να βρίσκει και να εκτυπώνει τις συντεταγμένες της διασταύρωσης που έχει τις περισσότερες καφετέριες σε απόσταση το πολύ d τετράγωνα από αυτήν (συνολική απόσταση μετακίνησης από τη διασταύρωση σε καφετέρια κατά μήκος δρόμων). Αν υπάρχουν περισσότερες από μία διασταύρωση με τον ίδιο μέγιστο αριθμό καφετεριών στη γειτονιά τους, το πρόγραμμά σας να βρίσκει την πιο νότια από αυτές. Μεταξύ εξ ίσου νότιων διασταυρώσεων, προτιμάται η πιο ανατολική.

   Μία ενδεικτική εκτέλεση του προγράμματος και ένα σχηματικό διάγραμμα των δεδομένων της εκτέλεσης φαίνονται στη συνέχεια.

```
% ./coffee 7 12 3
Please, give the coffee-shops map
..cc...c.cc.
....cc...c..
c..c....c...
............
..ccc....cc.
c.....c...cc
.....cccc...
Best junction at line 6 column 9 with 9 coffee-shops within 3 blocks distance
```



   Για τα δεδομένα της ενδεικτικής εκτέλεσης, υπάρχουν δύο διασταυρώσεις με τον ίδιο μέγιστο αριθμό καφετεριών στην περιοχή τους (σε απόσταση μέχρι 3 τετράγωνα), αυτές που σημειώνονται στο σχήμα με το X, αλλά η επιθυμητή λύση είναι η πιο νότια.

[2] Υποθέστε ότι όλα τα σπίτια στην πόλη βρίσκονται σε διασταυρώσεις δρόμων.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2014 (16/9/2014)

1. (α΄) "*Οι εντολές* `*p = x;` *και* `p = &x;` *στη C έχουν ακριβώς την ίδια λειτουργικότητα.*" Συμφωνείτε ή διαφωνείτε και γιατί;

   (β΄) Δίνονται τα παρακάτω τμήματα προγράμματος Α, Β και Γ.

```
signed char x = 1;
while (x++ > 0);
printf("%d\n", x);
```
Α

```
unsigned char x = 1;
while (x++ > 0);
printf("%d\n", x);
```
Β

```
unsigned char x = 1;
while (x++ >= 0);
printf("%d\n", x);
```
Γ

   Απαντήστε, για καθένα από αυτά, αν θα εκτυπώσει κάτι (και, αν ναι, τι;) ή θα εκτελείται επ΄ άπειρον. Δικαιολογήστε τις απαντήσεις σας.

   (γ΄) Τι θα εκτυπωθεί από τη συνάρτηση που δίνεται δεξιά, αν κληθεί δίνοντας στην είσοδο τους παρακάτω αριθμούς και γιατί;

```
void handle()
{  int x;
   scanf("%d", &x);
   if (!x)
      return;
   handle();
   printf("%d ", x);
}
```

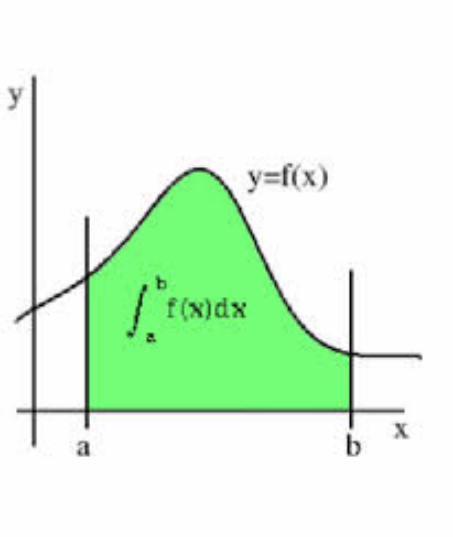
   `12 43 -356 7 81 -54 44 12 81 0`

   Δώστε μία εναλλακτική υλοποίηση της συνάρτησης `handle()`, στην οποία **να μην χρησιμοποιείται αναδρομή**, που να έχει την ίδια ακριβώς λειτουργικότητα με αυτή που δίνεται. Η νέα έκδοση της συνάρτησης που θα γράψετε θα πρέπει να είναι σε θέση να λειτουργεί σωστά, οσοδήποτε μεγάλη και να είναι η είσοδος που της δίνεται (με μόνο περιορισμό, ίσως, τη μνήμη του υπολογιστή σας), χωρίς όμως να κάνει αλόγιστη σπατάλη μνήμης.

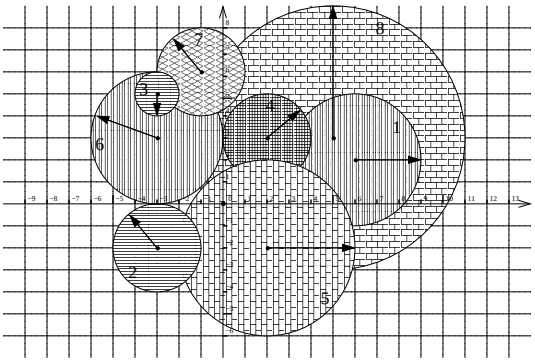
2. Στη συνέχεια δίνεται ο ορισμός μίας συνάρτησης `int ordtree(Treeptr)`, η οποία με τη βοήθεια μίας δεύτερης συνάρτησης, της οποίας ο ορισμός είναι ημιτελής, για δεδομένο δυαδικό δέντρο ακεραίων που της δίνεται ως όρισμα, επιστρέφει 1 ή 0, ανάλογα αν το δέντρο είναι ταξινομημένο (σε αύξουσα σειρά) ή όχι. Συμπληρώστε κατάλληλα τα `.....GAP X.....` της δεύτερης συνάρτησης (`X` = `A`, `B`, `C`, `D`, `E`, `F`, `G`, `H`), ώστε η αρχική να έχει την επιθυμητή λειτουργικότητα. Ο ορισμός του `Treeptr` είναι αυτός των σημειώσεων (`typedef` δείκτη σε κόμβο δυαδικού δέντρου ακεραίων, με μέλη `value`, `left` και `right`).

```
int ordtree(Treeptr p)
{  int a, b;
   return (p == NULL || fun(p, &a, &b)); }

int fun(Treeptr p, .....GAP A.....)
{  int a, b;
   if (p->left == NULL && .....GAP B.....) {
      *ap = p->value;
      .....GAP C..... }
   if (p->right == NULL) {
      *bp = p->value;
      return (fun(.....GAP D.....) && p->value > b); }
   if (.....GAP E.....) {
      .....GAP F.....
      return (fun(p->right, &a, bp) && GAP .....G.....); }
   return (.....GAP H.....); }
```

3. Γνωρίζετε από τα μαθηματικά ότι το ορισμένο ολοκλήρωμα $\int_a^b f(x)dx$ ισούται με το εμβαδόν της περιοχής που περικλείεται μεταξύ της γραφικής παράστασης της συνάρτησης $f(x)$, του άξονα x και των ευθειών $x = a$ και $x = b$, όπως φαίνεται στο διπλανό σχήμα. Ορίστε το πρωτότυπο συνάρτησης C με όνομα `defint`, και στη συνέχεια υλοποιήστε την, η οποία να δέχεται σαν ορίσματα δύο `double`, `a` και `b`, μία συνάρτηση, που δέχεται ένα όρισμα `double` και επιστρέφει `double`, και η οποία να υπολογίζει το προηγούμενο ορισμένο ολοκλήρωμα με ακρίβεια τουλάχιστον `epsilon` (`double` που επίσης δέχεται σαν όρισμα η `defint`).



4. Έστω ότι είναι δεδομένοι N κύκλοι σ΄ ένα ορθογώνιο σύστημα συντεταγμένων, όπως φαίνεται στο διπλανό σχήμα. Κάθε κύκλος καθορίζεται πλήρως από τις συντεταγμένες του κέντρου του και από το μήκος της ακτίνας του, τα οποία είναι πραγματικοί αριθμοί. Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`overlcircles.c`"), το οποίο αρχικά να δέχεται από τη γραμμή εντολής έναν ακέραιο N (πλήθος κύκλων). Στη συνέχεια, το πρόγραμμα να διαβάζει από την πρότυπη είσοδο (stdin) τα δεδομένα για N κύκλους, δηλαδή, για καθένα από αυτούς, τρεις `double` αριθμούς, τις συντεταγμένες (x, y) του κέντρου του και το μήκος της ακτίνας του r. Το πρόγραμμα να εκτυπώνει όλα τα ζευγάρια κύκλων που τέμνονται σε δύο σημεία (όχι απλά να εφάπτονται, εξωτερικά ή εσωτερικά). Ένα παράδειγμα εκτέλεσης φαίνεται στη συνέχεια:




```
% ./overlcircles 8
 6.0  2.0  3.0
-3.0 -2.0  2.0
-3.0  5.0  1.0
 2.0  3.0  2.0
 2.0 -2.0  4.0
-3.0  3.0  3.0
-1.0  6.0  2.0
 5.0  3.0  6.0
Circles 1 and 4 intersect in two points
Circles 1 and 5 intersect in two points
Circles 2 and 5 intersect in two points
Circles 3 and 7 intersect in two points
Circles 4 and 5 intersect in two points
Circles 5 and 8 intersect in two points
Circles 6 and 7 intersect in two points
Circles 6 and 8 intersect in two points
Circles 7 and 8 intersect in two points
%
```

Αν σας ζητείτο να υπολογίσετε, για τις περιπτώσεις κύκλων που τέμνονται σε δύο σημεία, και το εμβαδόν του κοινού τους τμήματος, περιγράψτε πώς θα το αντιμετωπίζατε, χωρίς να δώσετε κάποια υλοποίηση. Μπορείτε να προτείνετε είτε κάποια γεωμετρική μέθοδο, είτε κάποια αριθμητική/προσεγγιστική.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β′ Περιόδου 2015 (8/9/2015)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Ο συνδέτης επεξεργάζεται ...
   - ◯ Α. ... πηγαία και αντικειμενικά αρχεία
   - ◯ Β. ... αντικειμενικά αρχεία και βιβλιοθήκες
   - ◯ Γ. ... αντικειμενικά και εκτελέσιμα αρχεία

2. Ποιον τύπο μεταβλητής πρέπει να επιλέξουμε για την αναπαράσταση στη C του πλήθους των μαθητών μίας τάξης;
   - ◯ Α. `unsigned int`
   - ◯ Β. `double`
   - ◯ Γ. `int *`

3. Ποια είναι η τιμή της αριθμητικής παράστασης `(50 + 10.0) / 40` στη C;
   - ◯ Α. `1`
   - ◯ Β. `1.0`
   - ◯ Γ. `1.5`

4. Με ποια τιμή είναι ισοδύναμη μία αληθής λογική συνθήκη στη C;
   - ◯ Α. `1`
   - ◯ Β. `0`
   - ◯ Γ. `1.0`

5. Τι θα εκτυπώσει το παρακάτω τμήμα κώδικα C;

```
unsigned char x = 1;
x = x + x  ;  x = x * x  ;  x = x ^ x  ;
printf("%u\n", x);
```

   - ◯ Α. `0`
   - ◯ Β. `1`
   - ◯ Γ. `256`

6. Όταν σ′ ένα πρόγραμμα C διαβάζουμε επαναλαμβανόμενα χαρακτήρες από την είσοδο με τη συνάρτηση `getchar` και θέλουμε να αναγνωρίσουμε πότε φτάσαμε στο τέλος της εισόδου, τότε σε τι τύπου μεταβλητή πρέπει να αναθέσουμε αυτό που επιστρέφει η `getchar`;
   - ◯ Α. `char`
   - ◯ Β. `unsigned char`
   - ◯ Γ. `int`

7. Τι θα εκτυπώσει το παρακάτω τμήμα κώδικα C;

```
int x = 5, y = 7;
if (x = 5)
   if (y = 4)
      printf("alpha\n");
   else
      printf("beta\n");
else
   printf("gamma\n");
```

   - ◯ Α. `alpha`
   - ◯ Β. `beta`
   - ◯ Γ. `gamma`

8. Οι πίνακες στη C είναι προτιμότερο να ορίζονται ...
   ◯ A. ... στατικά
   ◯ B. ... δυναμικά
   ◯ Γ. ... στατικά ή δυναμικά, ανάλογα με την περίπτωση

9. Κατά την εκτέλεση ενός προγράμματος C, στο σωρό φυλάσσονται οι ...
   ◯ A. ... εξωτερικές μεταβλητές
   ◯ B. ... αυτόματες μεταβλητές
   ◯ Γ. ... μεταβλητές για τις οποίες έχει δεσμευτεί χώρος δυναμικά

10. Ο προσδιοριστής `static` σε μία εξωτερική μεταβλητή της C χρησιμοποιείται όταν θέλουμε ...
    ◯ A. ... να δηλώσουμε ότι η μεταβλητή δεν θα αλλάζει τιμή
    ◯ B. ... να ισχύει ο ορισμός της μεταβλητής για όλα τα αρχεία του προγράμματος
    ◯ Γ. ... να ισχύει ο ορισμός της μεταβλητής μόνο για το συγκεκριμένο αρχείο στο οποίο δηλώνεται

11. Για να διατηρεί μία μεταβλητή την τιμή της μεταξύ των κλήσεων μίας συνάρτησης C, πρέπει να οριστεί με τον προσδιοριστή ...
    ◯ A. ... `static`
    ◯ B. ... `local`
    ◯ Γ. ... `extern`

12. Στο τέλος της τελευταίας `case` μίας εντολής `switch` στη C ...
    ◯ A. ... απαγορεύεται να υπάρχει εντολή `break`, αλλιώς έχουμε συντακτικό λάθος
    ◯ B. ... είναι υποχρεωτικό να υπάρχει εντολή `break`, αλλιώς έχουμε συντακτικό λάθος
    ◯ Γ. ... δεν είναι απαραίτητο να υπάρχει εντολή `break`, αλλά είναι καλό να τη βάλουμε

13. Έστω ο δισδιάστατος δυναμικά δεσμευμένος πίνακας `array` τύπου `int**` και διαστάσεων `N x M` στη C. Ποια είναι η ισοδύναμη παράσταση για το `array[i][j]`;
    ◯ A. `*(*(array+i) + j)`
    ◯ B. `*(array + i*N + j)`
    ◯ Γ. `*(array + i*M + j)`

14. Μία δομή επανάληψης `do ... while` στη C ...
    ◯ A. ... μπορεί να μην κάνει καμία επανάληψη
    ◯ B. ... θα κάνει τουλάχιστον μία επανάληψη
    ◯ Γ. ... θα κάνει τουλάχιστον δύο επαναλήψεις

15. Η μακροεντολή της C

```
#define CUBE(X)      X * X * X
```

    για τον υπολογισμό του αποτελέσματος της ύψωσης ενός αριθμού στην τρίτη δύναμη ...
    ◯ A. ... είναι συντακτικά λάθος
    ◯ B. ... λειτουργεί με τον επιθυμητό τρόπο
    ◯ Γ. ... είναι συντακτικά σωστή, αλλά δεν λειτουργεί με τον επιθυμητό τρόπο

16. Τι θα εκτυπώσει το παρακάτω τμήμα κώδικα C;

```
unsigned char x = 8;
x = x + x  ;  x = x * x  ;
printf("%u\n", x);
```

    ◯ A. 0
    ◯ B. 1
    ◯ Γ. 256

17. Τι θα εκτυπώσει το παρακάτω τμήμα κώδικα C;

```
unsigned char x = 149, y = 227;
x = x ^ y  ;  y = x ^ y  ;  x = x ^ y  ;
printf("%u %u ", x, y);
x = x & y  ;  y = x | y  ;  x = ~ y  ;
printf("%u %u\n", x, y);
```

    ◯ A. 227 149 149 227
    ◯ B. 227 149 106 149
    ◯ Γ. Τίποτα από τα προηγούμενα δύο

18. Τι επιστρέφει η κλήση συνάρτησης `strcmp("abc","abc")` στη C;
    - ◯ Α. `1`
    - ◯ Β. `0`
    - ◯ Γ. `-1`

19. Τι θα εκτυπώσει το παρακάτω πρόγραμμα C;

```
void f(int x)
{ if (!x) return;
  printf("%d ", x);
  f(x-1); }

int main(void)
{  f(9);
   printf("\n");
   return 0; }
```

    - ◯ Α. ... 9 8 7 6 5 4 3 2 1 0
    - ◯ Β. ... 1 2 3 4 5 6 7 8 9
    - ◯ Γ. ... Τίποτα από τα προηγούμενα δύο

20. Έστω ο δισδιάστατος δυναμικά δεσμευμένος πίνακας `array` τύπου `int**` και διαστάσεων `N x M` στη C. Ποιος είναι ο ενδεδειγμένος κώδικας για την αποδέσμευση της μνήμης που καταλαμβάνει;
    - ◯ Α.
```
free(array);
```
    - ◯ Β.
```
for (i = 0 ; i < N ; ++i)
   free(array[i]);
free(array);
```
    - ◯ Γ.
```
free(array);
for (i = 0 ; i < N ; ++i)
   free(array[i]);
```

21. Το πλήθος των κόμβων μίας λίστας στη C ...
    - ◯ Α. ... ορίζεται στατικά μέσα στο πρόγραμμα
    - ◯ Β. ... ορίζεται δυναμικά κατά την εκτέλεση του προγράμματος
    - ◯ Γ. ... είναι δυνατόν να μεταβάλλεται κατά την εκτέλεση του προγράμματος

22. Τι θα εκτυπωθεί από την εντολή `./myargc good luck` αν το εκτελέσιμο πρόγραμμα `myargc` έχει προκύψει από το παρακάτω πρόγραμμα C;

```
int main(int argc, char **argv)
{  printf("%d\n", argc);
   return 0; }
```

    - ◯ Α. `1`
    - ◯ Β. `2`
    - ◯ Γ. `3`

23. Η έκφραση `*p->x++` στη C είναι ισοδύναμη με την ...
    - ◯ Α. ... `*(p->(x++))`
    - ◯ Β. ... `((*p)->x)++`
    - ◯ Γ. ... `*((p->x)++)`

24. Ένα δυαδικό δέντρο είναι δομή δεδομένων όπου κάθε κόμβος της ...
    - ◯ Α. ... έχει το πολύ δύο απογόνους παιδιά
    - ◯ Β. ... έχει ακριβώς δύο απογόνους παιδιά
    - ◯ Γ. ... έχει τουλάχιστον δύο απογόνους παιδιά

25. Οι μέσες πολυπλοκότητες των αλγορίθμων της φυσαλίδας και της εισαγωγής για την ταξινόμηση πινάκων είναι ...
    - ◯ Α. ... ίδιες
    - ◯ Β. ... διαφορετικές, με αυτή της φυσαλίδας να είναι καλύτερη
    - ◯ Γ. ... διαφορετικές, με αυτή της εισαγωγής να είναι καλύτερη

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Γράψτε πρόγραμμα C το οποίο θα δέχεται στη γραμμή εντολής έναν ακέραιο N μεγαλύτερο του 1 και θα εκτυπώνει στην έξοδο όλους τους *ελαττωματικούς* (deficient) αριθμούς που είναι σύνθετοι (όχι πρώτοι) και μικρότεροι ή ίσοι του N. Ένας αριθμός είναι ελαττωματικός όταν το άθροισμα των διαιρετών του, εξαιρουμένου του εαυτού του, είναι μικρότερο του αριθμού. Ένας αριθμός είναι πρώτος όταν έχει σαν μόνους διαιρέτες το 1 και τον εαυτό του. Ο αλγόριθμος που θα χρησιμοποιήσετε πρέπει να έχει πολυπλοκότητα αυστηρά καλύτερη από $O(N^2)$. Μία ενδεικτική εκτέλεση είναι η εξής:

```
$ deficientcomp 50
4 8 9 10 14 15 16 21 22 25 26 27 32 33 34 35 38 39 44 45 46 49 50
```

Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Τι θα εκτυπωθεί στην έξοδο από το παρακάτω πρόγραμμα; Δικαιολογήστε την απάντησή σας.

```
#include <stdio.h>

void f(int a, int *q, int **r)
{  printf(" %2d", ++a);
   *(q+1) = *q;
   *r = *r+2; }

void pr(int *x, int n)
{  int i;
   for (i = 0 ; i < n ; i++)
      printf(" %2d", x[i]);
   printf("\n"); }

int main(void)
{  int i, n, *p, x[] = {0,10,20,30,40,50,60,70,80,90};
   n = sizeof(x)/sizeof(int);
   p = x;
   for (i = 0 ; i < n/2 ; i++) {
      f(*p, p, &p);
      pr(x, n); }
   return 0; }
```

Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Υπάρχουν θετικοί ακέραιοι για τους οποίους ισχύει ότι αν δοθούν σαν είσοδος στο πρόγραμμα που φαίνεται στη συνέχεια αριστερά, αυτό δεν τερματίζει. Αιτιολογήστε το προηγούμενο, δίνοντας ένα παράδειγμα θετικού ακεραίου για τον οποίο το πρόγραμμα δεν τερματίζει. Συμπληρώστε, στη συνέχεια δεξιά, την επαναδιατύπωση της συνάρτησης `main()` του προγράμματος, χωρίς να αλλάξετε τη συνάρτηση `next()`, έτσι ώστε το νέο πρόγραμμα να συμπεριφέρεται ακριβώς με τον ίδιο τρόπο για εισόδους `X` που το αρχικό τερμάτιζε, δηλαδή να εκτυπώνει "`X is happy`", αλλά για εισόδους που το αρχικό δεν τερμάτιζε, το νέο να τερματίζει και να εκτυπώνει "`X is unhappy`".

```c
#include <stdio.h>

int next(int x)
{
    int sum = 0;
    while (x) {
        int digit = x % 10;
        sum += digit * digit;
        x /= 10;
    }
    return sum;
}

int main(void)
{
    int x;
    scanf("%d", &x);
    printf("%d is ", x);
    do {
        x = next(x);
    } while (x != 1);
    printf("happy\n");
    return 0;
}
```

```c
#include <stdio.h>

int next(int x)
{
    int sum = 0;
    while (x) {
        int digit = x % 10;
        sum += digit * digit;
        x /= 10;
    }
    return sum;
}

int main(void)
{
    int x;


    scanf("%d", &x);
    printf("%d is ", x);









    if (              )
        printf("happy\n");
    else
        printf("unhappy\n");
    return 0;
}
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2016 (21/9/2016)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Πότε ένα πρόγραμμα C που έχει συντακτικά λάθη (όχι προειδοποιήσεις) προκαλεί λάθη και κατά την εκτέλεση;
   - ◯ Α. Πάντοτε
   - ◯ Β. Μερικές φορές, αν ο μεταγλωττιστής αγνοήσει τα λάθη αυτά
   - ◯ Γ. Ποτέ, γιατί δεν έχει δημιουργηθεί εκτελέσιμο πρόγραμμα

2. Τα αρχεία με επέκταση `.h` είναι ...
   - ◯ Α. ... βιβλιοθήκες
   - ◯ Β. ... αρχεία επικεφαλίδας
   - ◯ Γ. ... αντικειμενικά αρχεία

3. Η συνάρτηση `rand` της C ...
   - ◯ Α. ... επιστρέφει έναν πραγματικό αριθμό στο διάστημα `[0,1]`
   - ◯ Β. ... επιστρέφει έναν ακέραιο αριθμό στο διάστημα `[0,RAND_MAX]`
   - ◯ Γ. ... αρχικοποιεί τη γεννήτρια τυχαίων αριθμών

4. Με ποια έκφραση είναι ισοδύναμη η `ch = getchar() != EOF` στη C;
   - ◯ Α. `(ch = getchar()) != EOF`
   - ◯ Β. `ch = (getchar() != EOF)`
   - ◯ Γ. `ch == getchar() != EOF`

5. Αν η `x` είναι ακέραια μεταβλητή, ποια τιμή θα πάρει μετά την εκτέλεση της εντολής `x = ((4 == 5) == 0);`
   - ◯ Α. `0`
   - ◯ Β. `1`
   - ◯ Γ. `-1`

6. Αν η `x` είναι τύπου `unsigned char`, ποια τιμή θα πάρει μετά την εντολή `x = ~((59 >> 4) & 0135);`
   - ◯ Α. `254`
   - ◯ Β. `252`
   - ◯ Γ. `249`

7. Πού φυλάσσονται οι τιμές των μεταβλητών που ορίζονται εντός των συναρτήσεων και έχουν δηλωθεί ως `static`;
   - ◯ Α. Στη στοίβα
   - ◯ Β. Στον σωρό
   - ◯ Γ. Στον εξωτερικό χώρο

8. Τι εκτυπώνεται από την παρακάτω εντολή;

```
printf("%d\n", printf("printf"));
```

   - ◯ Α. `printf6`
   - ◯ Β. `6printf`
   - ◯ Γ. `printf7`

9. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
unsigned char c = 0xFF;
printf("%u\n", c);
```

   - ◯ Α. `255`
   - ◯ Β. `1`
   - ◯ Γ. `-1`

10. Με τη δήλωση `int *f(void);` στη C ορίζεται το `f` ως ...
    - ◯ Α. ... συνάρτηση χωρίς ορίσματα που επιστρέφει δείκτη σε `int`
    - ◯ Β. ... μεταβλητή που είναι δείκτης σε συνάρτηση χωρίς ορίσματα που επιστρέφει δείκτη σε `int`
    - ◯ Γ. ... μεταβλητή που είναι δείκτης σε συνάρτηση χωρίς ορίσματα που επιστρέφει `int`

11. Η συνάρτηση `atoi` της C ...
    - ◯ Α. ... επιστρέφει για έναν ακέραιο `int` τη συμβολοσειρά που αποτελείται από τα ψηφία του ακεραίου
    - ◯ Β. ... επιστρέφει για μία αριθμητική συμβολοσειρά τον ακέραιο `int` που αντιστοιχεί σ΄ αυτήν
    - ◯ Γ. ... επιστρέφει όλα τα ψηφία ενός ακεραίου `int` ως συμβολοσειρές

12. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int i = 0;
for ( ; i <= 3 ; )
   printf("%d", ++i);
```

    - ◯ Α. 1234
    - ◯ Β. 0123
    - ◯ Γ. 123

13. Όταν η συνάρτηση `main` ενός προγράμματος C ορίζεται με ορίσματα, τότε ...
    - ◯ Α. ... πρέπει υποχρεωτικά τα ονόματα των τυπικών παραμέτρων της να είναι κατά σειρά τα `argc`, τύπου `int`, και `argv`, τύπου `char **` (ή πίνακας από `char *`)
    - ◯ Β. ... αρκεί η πρώτη τυπική παράμετρος να είναι τύπου `int` και η δεύτερη τύπου `char **` (ή πίνακας από `char *`), με οποιοδήποτε όνομα η κάθε μία
    - ◯ Γ. ... αρκεί η μία τυπική παράμετρος να είναι τύπου `int` και η άλλη τύπου `char **` (ή πίνακας από `char *`), με οποιαδήποτε σειρά και οποιοδήποτε όνομα η κάθε μία

14. Η τιμή μίας μεταβλητής τύπου `int *` ...
    - ◯ Α. ... απαιτεί λιγότερο χώρο για τη φύλαξή της από αυτόν της τιμής μίας μεταβλητής τύπου `int **`
    - ◯ Β. ... απαιτεί περισσότερο χώρο για τη φύλαξή της από αυτόν της τιμής μίας μεταβλητής τύπου `int **`
    - ◯ Γ. ... απαιτεί τον ίδιο χώρο για τη φύλαξή της με αυτόν της τιμής μίας μεταβλητής τύπου `int **`

15. Αν η μεταβλητή `p` στη C έχει ορισθεί ως δείκτης, τότε η έκφραση `p[-4]` ...
    - ◯ Α. ... είναι συντακτικά λανθασμένη
    - ◯ Β. ... είναι συντακτικά ορθή, αλλά σίγουρα θα προκαλέσει σφάλμα εκτέλεσης
    - ◯ Γ. ... είναι συντακτικά ορθή και ενδέχεται να έχει νόημα κατά την εκτέλεση

16. Τι θα εκτυπωθεί από την εντολή `./myargv good luck` αν το εκτελέσιμο πρόγραμμα `myargv` έχει προκύψει από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
int main(int argc, char **argv)
{  while (--argc)
      printf("%s ", *++argv);
   printf("\n");
   return 0; }
```

    - ◯ Α. `good luck`
    - ◯ Β. `./myargv good luck`
    - ◯ Γ. `g l`

17. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int *p, x[] = {1, 2, 3, 4, 5, 6};
p = x;
x[4] = *++p;
*p++ = x[5];
x[0] = (*p)++;
x[3] = ++(*p++);
*(p+2) = p[-1];
printf("%d %d %d %d %d %d\n", x[0], x[1], x[2], x[3], x[4], x[5]);
```

    - ◯ Α. 3 6 5 4 2 6
    - ◯ Β. 3 6 5 5 2 5
    - ◯ Γ. Τίποτα από τα προηγούμενα δύο, γιατί θα προκύψει σφάλμα εκτέλεσης

18. Τι θα εκτυπωθεί από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
void g(int);
void f(int x)
{  if (!x) return;
   printf("f%d ", x); g(x-1); }
void g(int x)
{  if (!x) return;
   printf("g%d ", x); f(x-1); }
int main(void)
{  f(6); printf("\n"); return 0; }
```

◯ A. `g0 f1 g2 f3 g4 f5`
◯ B. `f6 g5 f4 g3 f2 g1`
◯ Γ. `g1 f2 g3 f4 g5 f6`

19. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
char *str1 = "mystring", str2[10];
strcpy(str2, str1);
printf("%sequal\n", !strcmp(str1,str2) ? "" : "not ");
```

◯ A. `not equal`
◯ B. `equal`
◯ Γ. Τίποτα από τα προηγούμενα δύο, γιατί θα προκύψει σφάλμα εκτέλεσης

20. Με ποια έκφραση είναι ισοδύναμη η `p[i]` στη C;
◯ A. `*(p+i)`
◯ B. `*p+i`
◯ Γ. `*(*p+i)`

21. Οι συναρτήσεις `gets` και `fgets` της C παρουσιάζουν προβλήματα ασφάλειας. Συμφωνείτε;
◯ A. Απολύτως
◯ B. Η `gets` παρουσιάζει προβλήματα ασφάλειας, όχι όμως η `fgets`
◯ Γ. Η `fgets` παρουσιάζει προβλήματα ασφάλειας, όχι όμως η `gets`

22. Μία δομή ...
◯ A. ... δεν μπορεί να έχει ως μέλος της άλλη δομή
◯ B. ... μπορεί να έχει ως μέλος της άλλη δομή, οποιουδήποτε τύπου, ακόμα και ίδιου με αυτή
◯ Γ. ... μπορεί να έχει ως μέλος της δείκτη σε δομή οποιουδήποτε τύπου, ακόμα και ίδιου με αυτή

23. Σε ένα ταξινομημένο δυαδικό δέντρο, η ρίζα του ...
◯ A. ... είναι πάντα ο κόμβος με το μεγαλύτερο στοιχείο
◯ B. ... είναι πάντα ο κόμβος με το μικρότερο στοιχείο
◯ Γ. ... μπορεί να είναι οποιοσδήποτε κόμβος

24. Η μέση πολυπλοκότητα της σειριακής αναζήτησης σε πίνακα με n στοιχεία είναι ...
◯ A. ... $O(n^2)$
◯ B. ... $O(n \cdot logn)$
◯ Γ. ... $O(n)$

25. Η γρήγορη μέθοδος ταξινόμησης (quicksort) είναι πολύ γρηγορότερη αν ο πίνακας που πρόκειται να ταξινομηθεί είναι ήδη ταξινομημένος. Συμφωνείτε;
◯ A. Ναι
◯ B. Όχι
◯ Γ. Ναι, αν τα στοιχεία του πίνακα είναι μη μηδενικά

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Ορίζουμε ως *υπακολουθία* ενός αριθμού, οποιονδήποτε αριθμό προκύπτει με ψηφία από τον αρχικό αριθμό, όχι κατ΄ ανάγκη συνεχόμενα, αλλά με τη σειρά που εμφανίζονται στον αρχικό. Για παράδειγμα, όλες οι υπακολουθίες του 205 είναι οι 205, 20, 25, 2, 05, 0 και 5.

Γράψτε πρόγραμμα C το οποίο θα διαβάζει από την πρότυπη είσοδο ένα θετικό ακέραιο N και θα εκτυπώνει στην έξοδο όλες τις υπακολουθίες του N. Μία ενδεικτική εκτέλεση είναι η εξής:

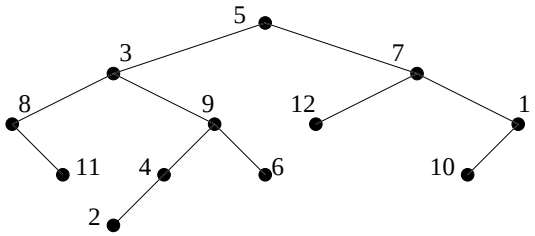
```
$ ./allsubseqs
Please, give a positive integer: 28408
All subsequences of 28408 are: 28408 2840 2848 284 2808 280 288 28 2408 240 248 24 208
20 28 2 8408 840 848 84 808 80 88 8 408 40 48 4 08 0 8
```

Παρατηρήστε στην ενδεικτική εκτέλεση ότι υπάρχουν στο αποτέλεσμα υπακολουθίες που εμφανίζονται παραπάνω από μία φορά. Αυτό είναι αναμενόμενο όταν υπάρχουν στον αριθμό επαναλαμβανόμενα ψηφία. Δεν χρειάζεται να φροντίσετε να μην εκτυπώνονται οι ταυτόσημες υπακολουθίες. Τις υπακολουθίες που αρχίζουν με 0 (ένα ή περισσότερα), επειδή υπάρχουν 0 στον αριθμό, μπορείτε να τις εκτυπώνετε είτε με τα 0 στην αρχή, είτε χωρίς αυτά. Δεν έχει σημασία. Επίσης, δεν έχει σημασία η σειρά με την οποία εκτυπώνονται οι υπακολουθίες.
Δεν επιτρέπεται η χρήση πινάκων (συμπεριλαμβανομένων και των συμβολοσειρών).

Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Στη C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου ακεραίων ως εξής:

```
typedef struct treenode *Trptr;

struct treenode {
  int v;
  Trptr left;
  Trptr right;
};
```



Τι θα εκτυπωθεί από τις συναρτήσεις **preord**, **inord** και **postord**, που δίνονται στη συνέχεια, αν κληθούν με όρισμα το δέντρο του παραπάνω σχήματος;

```
void preord(Trptr p)
{ if (p != NULL) {
    printf("%d ", p->v);
    preord(p->left);
    preord(p->right); }
}
```

```
void inord(Trptr p)
{ if (p != NULL) {
    inord(p->left);
    printf("%d ", p->v);
    inord(p->right); }
}
```

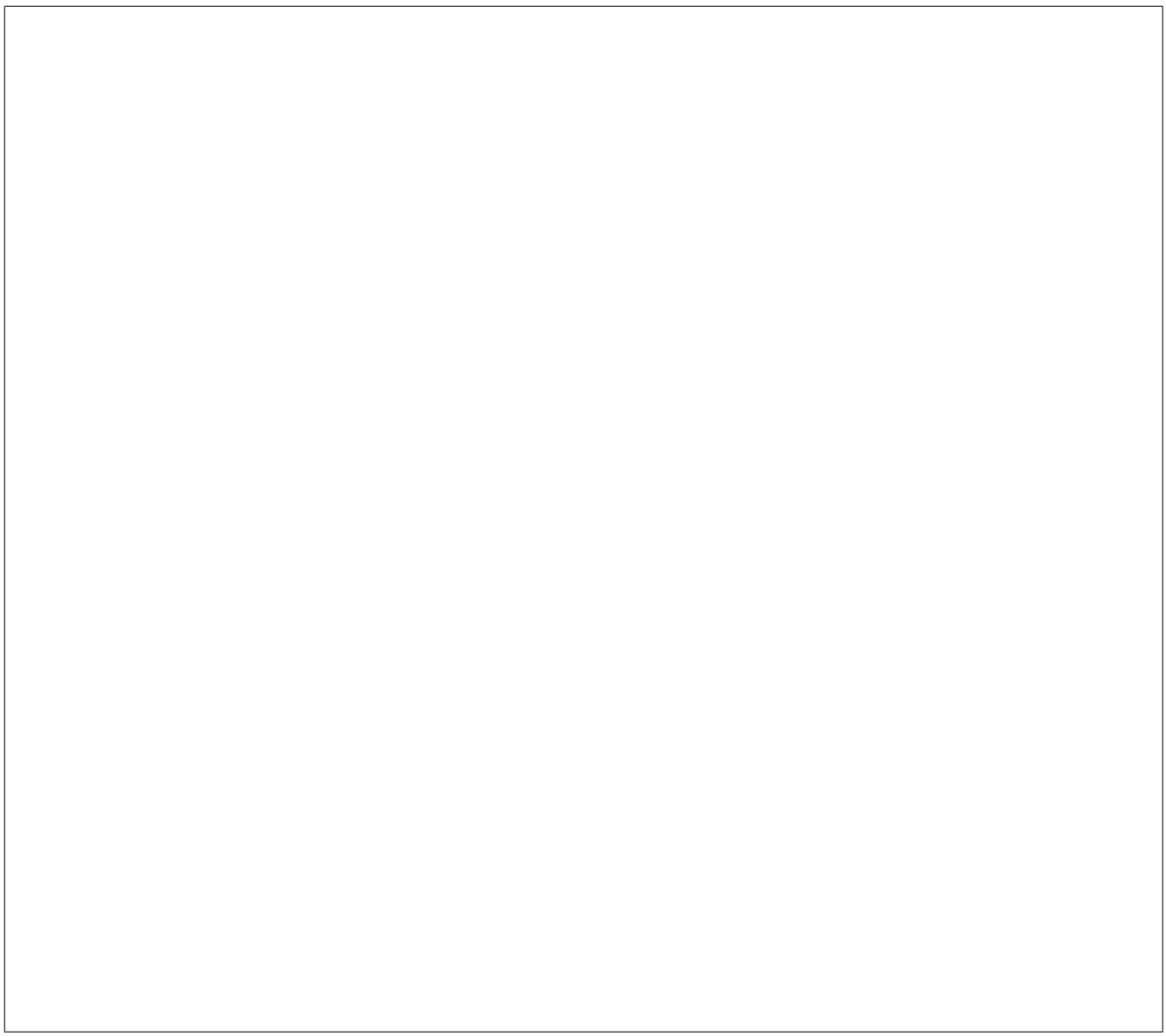
```
void postord(Trptr p)
{ if (p != NULL) {
    postord(p->left);
    postord(p->right);
    printf("%d ", p->v); }
}
```

Σχεδιάστε και ένα ταξινομημένο δυαδικό δέντρο ακεραίων (το παραπάνω δεν είναι ταξινομημένο) με 10 τουλάχιστον κόμβους και δώστε τις εκτυπώσεις των τριών συναρτήσεων και για το δέντρο αυτό. Έχετε να σχολιάσετε κάτι;

Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Γράψτε ένα πρόγραμμα C που να υπολογίζει όλα τα 2^{20} *mod* 3, 2^{200} *mod* 9, 2^{2000} *mod* 27, 2^{20000} *mod* 81, ..., $2^{2000000000}$ *mod* 19683, δηλαδή όλα τα $2^{2 \cdot 10^n}$ *mod* 3^n (υπόλοιπο διαίρεσης με το 3^n του 2 υψωμένου στη δύναμη 2 επί 10^n), για κάθε $n = 1, 2, 3, 4, \ldots, 9$. Παράδειγμα εκτέλεσης:

```
$ time ./compute_2_exp_2_times_10_exp_n_mod_3_exp_n
n = 1, mod = 1
n = 2, mod = 4
n = 3, mod = 4
n = 4, mod = 31
n = 5, mod = 139
n = 6, mod = 166
n = 7, mod = 1165
n = 8, mod = 2407
n = 9, mod = 11182
0.000u 0.000s 0:00.00 0.0%      0+0k 0+0io 0pf+0w
```



ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2017 (20/9/2017)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Ένα πρόγραμμα σε γλώσσα μηχανής εκτελείται στον ...
   - ◯ Α. ... προεπεξεργαστή
   - ◯ Β. ... επεξεργαστή κειμένου
   - ◯ Γ. ... επεξεργαστή

2. Η απόλυτη τιμή του μικρότερου ακέραιου που μπορεί να φυλαχθεί σε μεταβλητή τύπου `signed int` είναι ...
   - ◯ Α. ... μεγαλύτερη από ...
   - ◯ Β. ... μικρότερη από ...
   - ◯ Γ. ... ίση με ...

   ... την απόλυτη τιμή του μεγαλύτερου ακέραιου που μπορεί να φυλαχθεί στην ίδια μεταβλητή

3. Ο πραγματικός αριθμός `3.25` στο δεκαδικό σύστημα αρίθμησης μπορεί να αναπαρασταθεί στο δυαδικό ως ...
   - ◯ Α. ... `11.01`
   - ◯ Β. ... `11.11`
   - ◯ Γ. ... `10.01`

4. Ποιος από τους αριθμούς `B8D`, σε δεκαεξαδική μορφή, και `5615`, σε οκταδική μορφή, είναι μεγαλύτερος;
   - ◯ Α. Ο δεκαεξαδικός `B8D`
   - ◯ Β. Ο οκταδικός `5615`
   - ◯ Γ. Οι αριθμοί είναι ίσοι

5. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int y, x = 2;
if (x = 3)   y = x + 4;
else         y = x + 6;
printf("%d\n", y);
```

   - ◯ Α. 8
   - ◯ Β. 7
   - ◯ Γ. 6

6. Για να πολλαπλασιάσουμε μία `unsigned int` μεταβλητή με το 64, αρκεί να ...
   - ◯ Α. ... ολισθήσουμε τη δυαδική αναπαράσταση της τιμής της έξι θέσεις προς τα αριστερά, υπό την προϋπόθεση ότι τα bits που θα "χαθούν" θα είναι όλα μηδενικά
   - ◯ Β. ... ολισθήσουμε τη δυαδική αναπαράσταση της τιμής της έξι θέσεις προς τα αριστερά
   - ◯ Γ. ... ολισθήσουμε τη δυαδική αναπαράσταση της τιμής της έξι θέσεις προς τα δεξιά, υπό την προϋπόθεση ότι τα bits που θα "χαθούν" θα είναι όλα μηδενικά

7. Αν σε ένα πρόγραμμα C μία εξωτερική μεταβλητή έχει το ίδιο όνομα με μεταβλητή που έχει ορισθεί εντός μίας συνάρτησης, τότε ...
   - ◯ Α. ... μέσα στη συνάρτηση ισχύει η εξωτερική μεταβλητή
   - ◯ Β. ... μέσα στη συνάρτηση ισχύει η μεταβλητή που έχει ορισθεί εντός αυτής
   - ◯ Γ. ... το πρόγραμμα έχει συντακτικό λάθος

8. Ένα αρχείο επικεφαλίδας (`.h`) ...
   - ◯ Α. ... απαγορεύεται να περιέχει εκτελέσιμο κώδικα
   - ◯ Β. ... επιβάλλεται να περιέχει εκτελέσιμο κώδικα
   - ◯ Γ. ... ενδέχεται να περιέχει εκτελέσιμο κώδικα, αλλά δεν συνίσταται

9. Στη C, η πράξη `"358" + "1245"` ...
   - ◯ Α. ... δεν έχει κανένα νόημα και, μάλιστα, θα προκαλέσει συντακτικό λάθος
   - ◯ Β. ... έχει ως αποτέλεσμα το `"3581245"`
   - ◯ Γ. ... έχει ως αποτέλεσμα το `"1603"`

10. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
unsigned char x = 130;
x *= 2;
printf("%d\n", x);
```

◯ A. 260
◯ B. 4
◯ Γ. 5

11. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
#define sq(A) (A)*(A)
printf("%d\n", 80/sq(1+3));
```

◯ A. 86
◯ B. 5
◯ Γ. 80

12. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x[] = {7, 12}, y, *p;
p = x;
y = (*++p)++;
printf("%d %d %d\n", x[0], x[1], y);
```

◯ A. 7 13 12
◯ B. 7 12 13
◯ Γ. Τίποτα, γιατί θα προκληθεί σφάλμα κατά την εκτέλεση

13. Είναι δυνατόν με κατάλληλες οδηγίες προς τον προεπεξεργαστή να μην επιτρέψουμε τη μεταγλώττιση ενός τμήματος κάποιου προγράμματος C;
◯ A. Εξαρτάται από το πλήθος των γραμμών στο τμήμα προγράμματος
◯ B. Ναι, είναι δυνατόν
◯ Γ. Δεν είναι δυνατόν

14. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = -2, y = 8, z = 3, w;
w = (++x) + (y--) * (z++);
if (++x) w++;
printf("%d %d %d %d\n", x, y, z, w);
```

◯ A. -1 7 4 23
◯ B. 0 7 4 24
◯ Γ. 0 7 4 23

15. Ποια τιμή επιστρέφει η κλήση συνάρτησης `strlen("NULL")`;
◯ A. NULL
◯ B. 4
◯ Γ. 5

16. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int *p, x[] = {5, 6, 7, 8};
p = x+2;
x[3] = *p--;
*++p = x[0]--;
*(p-1) = x[1];
p += 8;
p[-10]--;
printf("%d %d %d %d\n", x[0], x[1], x[2], x[3]);
```

◯ A. 4 5 5 7
◯ B. 3 6 5 7
◯ Γ. Τίποτα από τις άλλες επιλογές

17. Αν η συνάρτηση `f()` είναι ορισμένη ως

```
void f(int x)
{  if (x < 3) {
      f(x+1); f(x+2); f(x+3); }
   else if (x == 3)
      printf("*"); }
```

τι θα εκτυπωθεί αν καλέσουμε το `f(0)`;
◯ Α. `***`
◯ Β. `****`
◯ Γ. `*****`

18. Η συνάρτηση `main()` ενός προγράμματος C επιστρέφει έναν ακέραιο που, κατά σύμβαση, ...
◯ Α. ... είναι το αποτέλεσμα του προγράμματος
◯ Β. ... δείχνει αν το πρόγραμμα τερμάτισε επιτυχώς
◯ Γ. ... είναι το πλήθος των συντακτικών λαθών που έχει το πρόγραμμα

19. Η συνάρτηση `strcpy()` της C ...
◯ Α. ... συγκρίνει δύο συμβολοσειρές
◯ Β. ... επιστρέφει το μήκος μίας συμβολοσειράς
◯ Γ. ... δημιουργεί αντίγραφο μίας συμβολοσειράς

20. Ένας δείκτης σε συνάρτηση μπορεί να χρησιμοποιηθεί ...
◯ Α. ... για την κλήση μίας συνάρτησης που θα αποφασισθεί ποια θα είναι κατά τη φάση της εκτέλεσης του προγράμματος
◯ Β. ... για τον βίαιο τερματισμό της εκτέλεσης μίας συνάρτησης
◯ Γ. ... για τη δυναμική τροποποίηση του κώδικα μίας συνάρτησης στην πηγαία του μορφή

21. Για να ανοίξουμε ένα αρχείο για ανάγνωση μόνο, αρκεί να καλέσουμε τη συνάρτηση ...
◯ Α. ... `fread()`
◯ Β. ... `fseek()`
◯ Γ. ... `fopen()`

22. Αν ένα εκτελέσιμο πρόγραμμα, που έχει προκύψει από το πηγαίο πρόγραμμα C που ακολουθεί, κληθεί ως `./myprog 5 7` τι θα εκτυπώσει;

```
#include <stdio.h>
int main(int argc, char **argv)
{  printf("%s ", *argv++);
   printf("%s\n", *++argv);
   return 0; }
```

◯ Α. `5 7`
◯ Β. `./myprog 5`
◯ Γ. `./myprog 7`

23. Τα μέλη τύπου δείκτη στη δομή ενός κόμβου απλά συνδεδεμένης λίστας είναι ...
◯ Α. ... ακριβώς ένα
◯ Β. ... ακριβώς δύο
◯ Γ. ... τουλάχιστον ένα

24. Σε ένα ταξινομημένο δυαδικό δέντρο ακεραίων, το περιεχόμενο της ρίζας του ...
◯ Α. ... έχει πάντοτε την ελάχιστη τιμή των περιεχομένων των κόμβων του δέντρου
◯ Β. ... έχει πάντοτε τιμή που βρίσκεται μέσα στο ανοικτό διάστημα που ορίζουν το πλέον αριστερό και το πλέον δεξιό φύλλο του δέντρου
◯ Γ. ... έχει πάντοτε τιμή που βρίσκεται μέσα στο κλειστό διάστημα που ορίζουν το πλέον αριστερό και το πλέον δεξιό φύλλο του δέντρου

25. Ο ταχύτερος αλγόριθμος για να αναζητήσουμε ένα στοιχείο μέσα σε ένα μη ταξινομημένο πίνακα έχει πολυπλοκότητα ...
◯ Α. ... $O(n \cdot logn)$
◯ Β. ... $O(n)$
◯ Γ. ... $O(logn)$

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

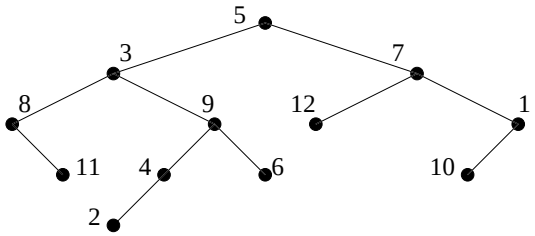
Γράψτε πρόγραμμα C το οποίο θα διαβάζει μία γραμμή από την πρότυπη είσοδο και θα βρίσκει τη μέγιστη ακολουθία από συνεχόμενους ίδιους χαρακτήρες. Το πρόγραμμα να εκτυπώνει το πλήθος των συνεχόμενων εμφανίσεων του ίδιου χαρακτήρα στη μέγιστη ακολουθία που βρήκε, τον κωδικό του χαρακτήρα, τον ίδιο τον χαρακτήρα και την αρχική και τελική θέση της ακολουθίας στη γραμμή που διαβάστηκε (η πρώτη θέση στη γραμμή είναι η 0). Αν υπάρχουν στην είσοδο δύο η περισσότερες εξ ίσου μέγιστες ακολουθίες από ίδιους χαρακτήρες, το πρόγραμμα να εκτυπώνει τα στοιχεία για την πρώτη κατά σειρά από αυτές. Για να βαθμολογηθεί η απάντηση που θα δώσετε, θα πρέπει το πρόγραμμά σας να λειτουργεί σωστά για οσοδήποτε μεγάλη είσοδο, ακόμα και αν αυτή δεν μπορεί να φυλαχθεί ολόκληρη στη μνήμη του υπολογιστή. Μία ενδεικτική εκτέλεση:

```
$  ./maxcontch
ab***ccadddd[1]eeeef3333

Maximum number of 4 continuous occurrences of character with code 100 ('d')
Start position in input is 8
  End position in input is 11
```

Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Στη C, συνήθως ορίζουμε τον κόμβο ενός δυαδικού δέντρου ακεραίων ως εξής:

```
typedef struct treenode *Treeptr;

struct treenode {
  int value;
  Treeptr left;
  Treeptr right;
};
```



Η συνάρτηση `myfun()` που φαίνεται δεξιά εκτυπώνει τους κόμβους του δυαδικού δέντρου που της δίνεται ως όρισμα με μία συγκεκριμένη σειρά. Με ποια σειρά θα εκτυπωθούν οι κόμβοι του παραπάνω δέντρου από τη συνάρτηση;

Παρατηρείτε ότι η συνάρτηση `myfun()` καλεί την `pl()` που είναι αναδρομική. Προτείνετε έναν αλγόριθμο, ή υλοποιήστε τη σχετική συνάρτηση σε C, αν προτιμάτε, που να έχει το ίδιο αποτέλεσμα με τη συνάρτηση `myfun()`, χωρίς όμως να χρησιμοποιεί αναδρομή.

```
void myfun(Treeptr p)
{  int lv = 0;
   while (pl(p, lv))
      lv++;
}

int pl(Treeptr p, int lv)
{  if (p == NULL) return 0;
   if (!lv) {
      printf("%d ", p->value);
      return 1;
   }
   return (pl(p->left, lv-1) | pl(p->right, lv-1));
}
```

Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Ορίστε δύο συναρτήσεις C με πρωτότυπα

```
unsigned int add(unsigned int a, unsigned int b)
```

και

```
unsigned int mult(unsigned int a, unsigned int b)
```

οι οποίες να επιστρέφουν το άθροισμα και το γινόμενο, αντίστοιχα, των ορισμάτων που τους δίνονται. Μία λεπτομέρεια είναι ότι δεν επιτρέπεται να χρησιμοποιήσετε στον κώδικα των συναρτήσεων τους χαρακτήρες `+`, `-`, `*`, `/`, `%`, `[` και `]`.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2018 (21/9/2018)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Στην εντολή `#include <filename>`, το `<filename>` πρέπει να είναι ...
   ◯ Α. ... βιβλιοθήκη
   ◯ Β. ... αρχείο επικεφαλίδας
   ◯ Γ. ... πηγαίο αρχείο

2. Υπό την προϋπόθεση ότι το πηγαίο αρχείο `aprog.c` δεν έχει συντακτικά λάθη, με την εντολή `gcc -c aprog.c` παράγεται ...
   ◯ Α. ... το αντικειμενικό αρχείο `aprog.o`
   ◯ Β. ... το εκτελέσιμο αρχείο `a.out`
   ◯ Γ. ... το εκτελέσιμο αρχείο `aprog`

3. Στο οκταδικό σύστημα αρίθμησης χρησιμοποιούνται ...
   ◯ Α. ... 7 ψηφία
   ◯ Β. ... 8 ψηφία
   ◯ Γ. ... 10 ψηφία

4. Για να έχει νόημα η εντολή `x = 176 >> 2.1;`, πώς πρέπει να έχει δηλωθεί η μεταβλητή x;
   ◯ Α. `int x;`
   ◯ Β. `double x;`
   ◯ Γ. Δεν έχει σημασία, γιατί, ούτως ή άλλως, η εντολή είναι συντακτικά λανθασμένη

5. Η εντολή `++x++;` ...
   ◯ Α. ... είναι ισοδύναμη με την ακολουθία εντολών `++x; x++;`
   ◯ Β. ... θα προκαλέσει λάθος κατά την εκτέλεσή της
   ◯ Γ. ... είναι συντακτικά λανθασμένη

6. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
signed char x = -1;
if (~x == x^x) printf("yes\n");
else           printf("no\n");
```

   ◯ Α. `yes`
   ◯ Β. `no`
   ◯ Γ. Το πρόγραμμα έχει συντακτικό λάθος και δεν μεταγλωττίζεται

7. Με τη συνάρτηση `realloc` μπορούμε να τροποποιήσουμε το μέγεθος της μνήμης που έχει δεσμευθεί για ένα στατικό πίνακα.
   ◯ Α. Σωστό
   ◯ Β. Λάθος
   ◯ Γ. Σωστό, αν ο πίνακας είναι μονοδιάστατος

8. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
double x = 2.8, y;
int z, w;
z = (int) x;
y = (double) z;
w = (int) (x * y);
printf("%d\n", w);
```

   ◯ Α. 6
   ◯ Β. 5
   ◯ Γ. 8

9. Τι θα εκτυπωθεί από την εκτέλεση της εντολής `printf("%d\n", 0111);`
   ◯ Α. `111`
   ◯ Β. `7`
   ◯ Γ. `73`

10. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = 3, y = 22;
if (x == 3) { if (y == 6) printf("one"); }
else printf("two");
printf("three\n");
```

   ◯ Α. `onetwothree`
   ◯ Β. `twothree`
   ◯ Γ. `three`

11. Η `main` συνάρτηση ενός προγράμματος ...
   ◯ Α. ... εκτυπώνει στην οθόνη ένα κωδικό εξόδου
   ◯ Β. ... επιστρέφει στο λειτουργικό σύστημα ένα κωδικό εξόδου
   ◯ Γ. ... δέχεται από κάθε συνάρτηση που καλεί ένα κωδικό εξόδου

12. Ποια είναι η σωστή δήλωση για τη μεταβλητή `fun`, ώστε να είναι συντακτικά σωστή η εντολή `fun = strcpy;`, όπου `strcpy` είναι η γνωστή συνάρτηση βιβλιοθήκης της C για την αντιγραφή μίας συμβολοσειράς σε μία άλλη;
   ◯ Α. `char (**fun)(char *, char *);`
   ◯ Β. `char *(*fun)(char *, char *);`
   ◯ Γ. `char **fun(char *, char *);`

13. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x, y = 5, z = 3, w;
x = ++y + z++;
w = x-- + --y * z++;
printf("%d\n", w);
```

   ◯ Α. `29`
   ◯ Β. `38`
   ◯ Γ. Τίποτα από τις άλλες επιλογές

14. Τι θα εκτυπωθεί από την παρακάτω εντολή;

```
printf("%c\n", (('t'-'T') == ('b'-'B')) + 'D');
```

   ◯ Α. `D`
   ◯ Β. `E`
   ◯ Γ. `e`

15. Έστω ότι το εκτελέσιμο πρόγραμμα που προκύπτει από το πηγαίο πρόγραμμα C που ακολουθεί ονομάζεται `myprog`. Αν αυτό κληθεί ως `./myprog abc de fghij klm`, τι θα εκτυπωθεί;

```
#include <stdio.h>
int main(int argc, char *argv[])
{  if (argc == 1) return 1;
   while (--argc)
      printf("%c ", *argv[argc]);
   printf("\n");
   return 0; }
```

   ◯ Α. `mlkjihgfedcba`
   ◯ Β. `k f d a`
   ◯ Γ. `klm fghij de abc`

16. Ένα πλεονέκτημα των πινάκων που ορίζονται δυναμικά, σε σχέση με τους στατικούς πίνακες, είναι ότι ...
   ◯ Α. ... η ταχύτητα προσπέλασης των στοιχείων τους είναι πολύ μεγαλύτερη
   ◯ Β. ... χρειάζονται λιγότερη μνήμη για την αποθήκευση ίδιου πλήθους δεδομένων
   ◯ Γ. ... μπορούμε να αλλάξουμε τη διάστασή τους κατά την εκτέλεση του προγράμματος

17. Η παράσταση `*++p->next` **δεν** είναι ισοδύναμη με την ...
    - ◯ Α. ... `*++(p->next)`
    - ◯ Β. ... `*(++p->next)`
    - ◯ Γ. ... `*(++p)->next`

18. Ο σωστός τρόπος για να διαβάσουμε από την είσοδο την τιμή μίας ακέραιας μεταβλητής `x` είναι:
    - ◯ Α. `scanf("%d", x);`
    - ◯ Β. `scanf("%d", &x);`
    - ◯ Γ. `scanf("%d", *x);`

19. Για να έχουμε πρόσβαση στους κόμβους μίας απλά συνδεδεμένης λίστας, ...
    - ◯ Α. ... χρειάζεται να γνωρίζουμε τη διεύθυνση του πρώτου κόμβου της
    - ◯ Β. ... χρειάζεται να γνωρίζουμε τις διευθύνσεις όλων των κόμβων της
    - ◯ Γ. ... χρειάζεται να γνωρίζουμε τη διεύθυνση του πρώτου κόμβου της και το πλήθος των κόμβων της

20. Τι θα εκτυπωθεί από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
void g(int);
void f(int x)
{  if (x < 1) return;
   g(x-1); }
void g(int x)
{  if (x > 2)
      { f(x-2); f(x-1);}
   printf("*"); }
int main(void)
{  f(6); printf("\n"); return 0; }
```

    - ◯ Α. `****`
    - ◯ Β. `*****`
    - ◯ Γ. `******`

21. Η εντολή `typedef` της C συνεισφέρει ...
    - ◯ Α. ... στην ταχύτητα εκτέλεσης των προγραμμάτων
    - ◯ Β. ... στη γρήγορη μεταγλώττιση των προγραμμάτων
    - ◯ Γ. ... στην αναγνωσιμότητα των προγραμμάτων

22. Ο πιο συνηθισμένος τρόπος υλοποίησης της συνάρτησης `int putchar(int)` της C είναι ...
    - ◯ Α. ... μέσω C
    - ◯ Β. ... μέσω assembly
    - ◯ Γ. ... μέσω μακροεντολής

23. Αν για ένα πρόβλημα υπάρχουν δύο αλγόριθμοι Α και Β, όπου οι πολυπλοκότητες χρόνου τους είναι $O((n \cdot logn)^2)$ και $O(n^3 \cdot logn)$, αντίστοιχα, τότε ...
    - ◯ Α. ... ο αλγόριθμος Α είναι καλύτερος από τον Β, για μεγάλες τιμές του n
    - ◯ Β. ... ο αλγόριθμος Β είναι καλύτερος από τον Α, για μεγάλες τιμές του n
    - ◯ Γ. ... οι δύο αλγόριθμοι είναι εξ ίσου καλοί, για οποιαδήποτε τιμή του n

24. Ποια είναι η πολυπλοκότητα της μεθόδου ταξινόμησης της συγχώνευσης;
    - ◯ Α. $O(n \cdot logn)$
    - ◯ Β. $O(n^2)$
    - ◯ Γ. $O(n^3)$

25. Πού θα γίνουν οι Ολυμπιακοί Αγώνες το 2020;
    - ◯ Α. Rio de Janeiro
    - ◯ Β. Tokyo
    - ◯ Γ. Qatar

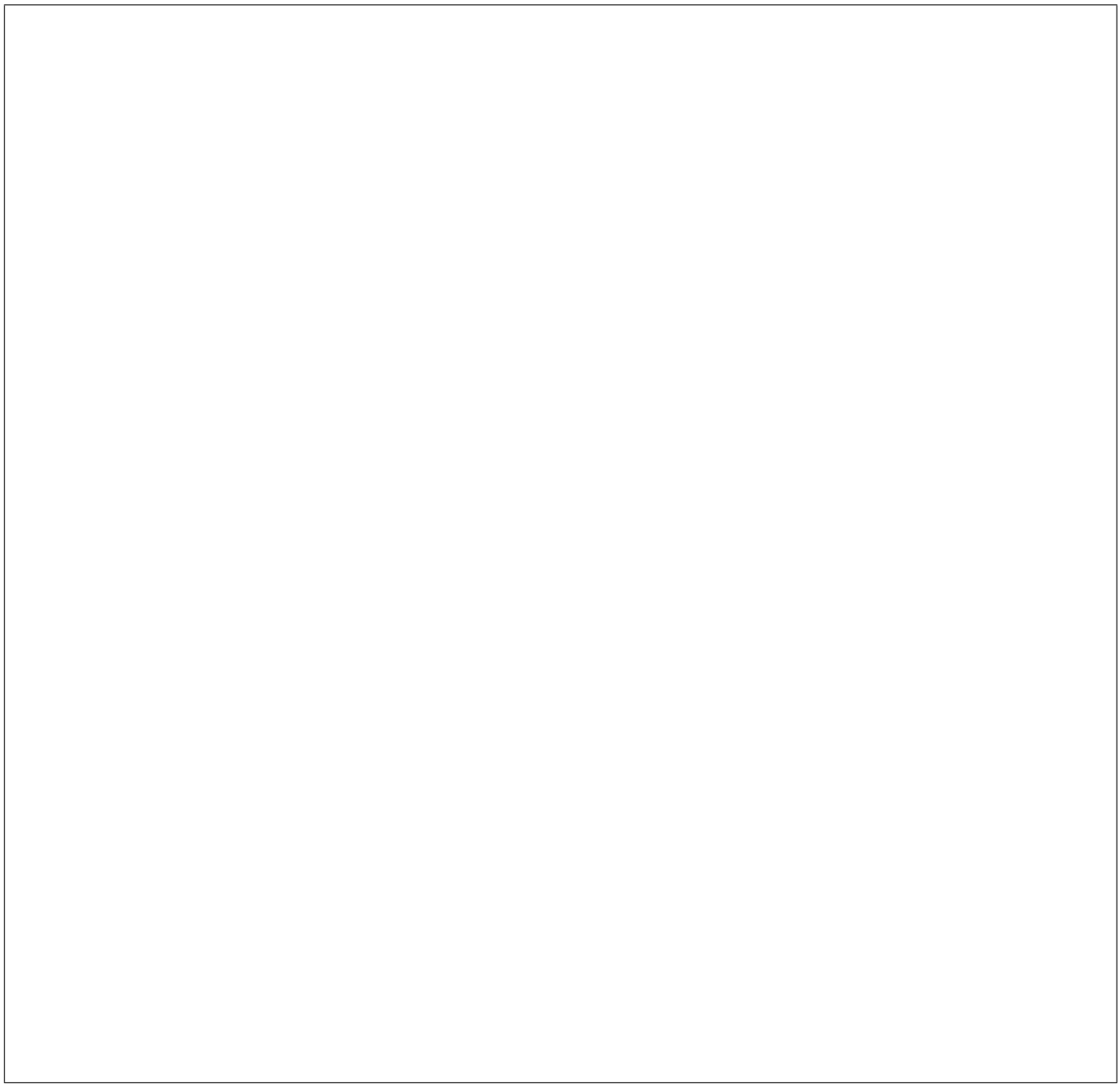
Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Η ακολουθία Recamán ορίζεται αναδρομικά ως εξής:

$$a_n = \begin{cases} 0, & \text{αν} \quad n = 0 \\ a_{n-1} - n, & \text{αν} \quad a_{n-1} - n > 0 \quad \text{και} \quad a_{n-1} - n \neq a_j \quad \text{για κάθε } j \text{ με} \quad 0 \leq j \leq n-1 \\ a_{n-1} + n, & \text{διαφορετικά} \end{cases}$$

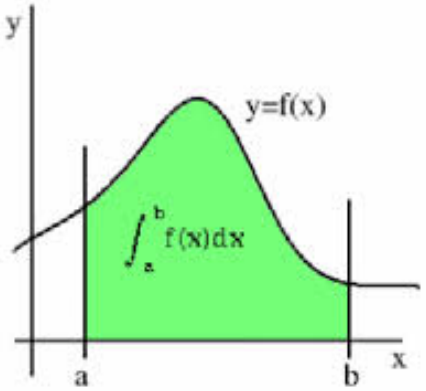
Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`recaman.c`"), το οποίο να παίρνει από τη γραμμή εντολής ένα θετικό ακέραιο N και να εκτυπώνει τους N πρώτους όρους της ακολουθίας Recamán. Μία ενδεικτική εκτέλεση:

```
$ ./recaman 100
0 1 3 6 2 7 13 20 12 21 11 22 10 23 9 24 8 25 43 62 42 63 41 18 42 17 43 16 44 15 45
14 46 79 113 78 114 77 39 78 38 79 37 80 36 81 35 82 34 83 33 84 32 85 31 86 30 87
29 88 28 89 27 90 26 91 157 224 156 225 155 226 154 227 153 228 152 75 153 74 154 73
155 72 156 71 157 70 158 69 159 68 160 67 161 66 162 65 163 64
```



Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Γνωρίζετε από τα μαθηματικά ότι το ορισμένο ολοκλήρωμα $\int_a^b f(x)dx$ ισούται με το εμβαδόν της περιοχής που περικλείεται μεταξύ της γραφικής παράστασης της συνάρτησης $f(x)$, του άξονα x και των ευθειών $x = a$ και $x = b$, όπως φαίνεται στο διπλανό σχήμα. Ορίστε το πρωτότυπο συνάρτησης C με όνομα `defint`, και στη συνέχεια υλοποιήστε την, η οποία να δέχεται σαν ορίσματα δύο `double`, `a` και `b`, μία συνάρτηση, που δέχεται ένα όρισμα `double` και επιστρέφει `double`, και η οποία να υπολογίζει το προηγούμενο ορισμένο ολοκλήρωμα με ακρίβεια τουλάχιστον `epsilon` (`double` που επίσης δέχεται σαν όρισμα η `defint`).



Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Τι πιστεύετε ότι θα εκτυπωθεί από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{  int i = 0, n;
   double x, y;
   for (n = 0 ; n < 1000000 ; n++)  {
      x = 2.0 * rand() / RAND_MAX - 1;
      y = 2.0 * rand() / RAND_MAX - 1;
      if (x * x + y * y < 1)
         i++;
   }
   printf("%.2f\n", 4.0 * i / n);
   return 0;
}
```

Αιτιολογήστε την απάντησή σας.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β′ Περιόδου 2019 (2/9/2019)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (50/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Ο προεπεξεργαστής της C δέχεται ως είσοδο πηγαία αρχεία C και παράγει ως έξοδο ...
   ◯ Α. ... κώδικα C
   ◯ Β. ... κώδικα assembly
   ◯ Γ. ... γλώσσα μηχανής

2. Το πρόγραμμα `Dev-C++` στα Windows είναι ...
   ◯ Α. ... μεταγλωττιστής
   ◯ Β. ... ταυτόχρονα μεταγλωττιστής και συνδέτης
   ◯ Γ. ... μία διεπαφή που, μεταξύ άλλων, τελικά καλεί ένα μεταγλωττιστή, όπως ο `gcc`

3. Όλοι οι πραγματικοί αριθμοί στο διάστημα `[0,1]` μπορούν να αναπαρασταθούν με απόλυτη ακρίβεια στον υπολογιστή.
   ◯ Α. Σωστό
   ◯ Β. Λάθος
   ◯ Γ. Σωστό, μόνο όταν ο υπολογιστής έχει αρχιτεκτονική 64-bit

4. Ένας πραγματικός αριθμός, οσοδήποτε μεγάλος και να είναι, μπορεί να αναπαρασταθεί στον υπολογιστή, έστω και κατά προσέγγιση, μέσω κάποιου από τους προκαθορισμένους τύπους μεταβλητών της C.
   ◯ Α. Σωστό
   ◯ Β. Λάθος
   ◯ Γ. Σωστό, μόνο όταν χρησιμοποιηθούν 16 bytes για την αποθήκευση

5. Το πλήθος των διαφορετικών τιμών που μπορεί να πάρει μία ακέραια μεταβλητή είναι πάντοτε δύναμη του 2, ανεξάρτητα από το πλήθος των bytes που χρησιμοποιούνται για την αποθήκευσή της.
   ◯ Α. Σωστά, μόνο όταν η μεταβλητή είναι `signed`
   ◯ Β. Σωστά, μόνο όταν η μεταβλητή είναι `unsigned`
   ◯ Γ. Σωστά, σε κάθε περίπτωση

6. Αν η `x` είναι δηλωμένη ως ακέραια μεταβλητή, τότε η εντολή `x + 5 = 8;`
   ◯ Α. ... έχει ως αποτέλεσμα να πάρει η μεταβλητή `x` την τιμή `3`
   ◯ Β. ... είναι συντακτικά λανθασμένη και δεν μεταγλωττίζεται
   ◯ Γ. ... μεταγλωττίζεται, αλλά θα προκαλέσει λάθος κατά την εκτέλεση

7. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
unsigned int x = 27;
printf("%u\n", ~x & 0xFF);
```

   ◯ Α. `228`
   ◯ Β. `-27`
   ◯ Γ. `100`

8. Αν οι ακέραιες μεταβλητές `x` και `y` έχουν τις τιμές `5` και `-5`, αντίστοιχα, ποια τιμή θα δοθεί στην ακέραια μεταβλητή `z` μετά την εκτέλεση της εντολής `z = x = y;`
   ◯ Α. `5`
   ◯ Β. `-5`
   ◯ Γ. `0`

9. Αν έχουμε δεσμεύσει με τη συνάρτηση `malloc` χώρο για ένα πίνακα, αλλά τελικά μας αρκεί ένα μικρό τμήμα του χώρου αυτού, μπορούμε εκ των υστέρων μέσω της συνάρτησης `free` να αποδεσμεύσουμε τον επιπλέον χώρο.
   ◯ Α. Σωστά
   ◯ Β. Λάθος
   ◯ Γ. Σωστά, αλλά μόνο όταν ο πίνακας είναι μονοδιάστατος

10. Αν η συμβολοσειρά στην οποία δείχνει η μεταβλητή `s1` προηγείται αλφαβητικά της συμβολοσειράς στην οποία δείχνει η μεταβλητή `s2`, ποια από τις εξής εντολές θα δώσει, σε κάθε περίπτωση, στη μεταβλητή `x` την τιμή 1;
    ◯ Α. `x = (s1 < s2);`
    ◯ Β. `x = !strcmp(s1,s2);`
    ◯ Γ. `x = strcmp(s1,s2);`

11. Με ποια από τις παρακάτω εντολές είναι ισοδύναμη, σε κάθε περίπτωση, η εντολή `char str[] = "Hello";`
    ◯ Α. `char str[6] = {'H','e','l','l','o','\0'};`
    ◯ Β. `char str[] = {'H','e','l','l','o'};`
    ◯ Γ. `char *str = "Hello";`

12. Η συνθήκη `x && y == x & y` ...
    ◯ Α. ... είναι πάντα αληθής, ανεξάρτητα από τις τιμές των μεταβλητών `x` και `y`
    ◯ Β. ... είναι σε κάποιες περιπτώσεις αληθής, ανάλογα με τις τιμές των μεταβλητών `x` και `y`
    ◯ Γ. ... ποτέ δεν μπορεί να είναι αληθής

13. Τι θα εκτυπωθεί από το παρακάτω τμήμα κώδικα C;

```
int x = 8, y = 1;
int z = 0;
while (x-- != ++y)
   z++;
printf("%d\n", z);
```

    ◯ Α. 3
    ◯ Β. 4
    ◯ Γ. Τίποτα, γιατί ο κώδικας θα εκτελείται επ΄ άπειρον

14. Αν το εκτελέσιμο πρόγραμμα που προκύπτει από το πηγαίο πρόγραμμα C που ακολουθεί (έστω με όνομα `myprog`) κληθεί ως `./myprog ab cde fghij`, τι θα εκτυπωθεί στην έξοδο;

```
#include <stdio.h>
int main(int argc, char **argv) {
   int z = 0;
   while (argv[++z] != NULL);
   printf("%d\n", z);
   return 0; }
```

    ◯ Α. 3
    ◯ Β. 4
    ◯ Γ. 5

15. Ποια θα είναι τα περιεχόμενα του πίνακα `x` μετά την εκτέλεση του παρακάτω τμήματος κώδικα C;

```
int *p;
int x[] = {5, 7, 2, 3, 6, 0, 1, 4};
p = x;
while (*p = *(p+2))
   p++;
```

    ◯ Α. 2 3 6 0 6 0 1 4
    ◯ Β. 7 2 3 6 0 0 1 4
    ◯ Γ. 2 3 6 0 1 4 0 0

16. Αν η `x` είναι μεταβλητή τύπου `unsigned int`, τότε η εντολή `x <<= 3;` είναι ισοδύναμη με την εντολή ...
    ◯ Α. ... `x = x / 3;`
    ◯ Β. ... `x /= 8;`
    ◯ Γ. ... `x = 8 * x;`

17. Αν η συνάρτηση `main` ενός προγράμματος C έχει ορισθεί ως `int main(int argc, char **argv)` και το πρόγραμμα κληθεί χωρίς ορίσματα, τότε η μεταβλητή `argc` θα έχει την τιμή ...
    ◯ Α. ... 0
    ◯ Β. ... 1
    ◯ Γ. ... -1

18. Ζητήθηκε από κάποιον να γράψει ένα πρόγραμμα C το οποίο θα υπολογίζει το άθροισμα (`sum`) των διαιρετών ενός ακεραίου αριθμού (`number`) μεγαλύτερου του 1 και το σχετικό τμήμα του κώδικά του ήταν το εξής:

```
for (sum = number + 1, divisor = 2 ; divisor * divisor <= number ; divisor++) {
   if (!(number % divisor))
      sum += divisor + number / divisor;
   if (divisor * divisor == number)
      sum -= divisor; }
```

◯ A. Ο κώδικας είναι σωστός
◯ B. Ο κώδικας δεν είναι σωστός, αλλά μπορεί να διορθωθεί αν στη συνθήκη τερματισμού του `for` αντικατασταθεί ο τελεστής `<=` με τον `<`
◯ Γ. Ο κώδικας δεν είναι σωστός, αλλά μπορεί να διορθωθεί με την προσθήκη επιπλέον εντολής (ή εντολών)

19. Τι θα εκτυπωθεί από το παρακάτω πρόγραμμα C;

```
#include <stdio.h>
int count;
void f(int n) {
   if (n > 4) return;
   count++;
   f(n+1); f(n+2); f(n+3); }
int main(void) {
   f(0);
   printf("%d \n", count);
   return 0; }
```

◯ A. 8
◯ B. 15
◯ Γ. Τίποτα, γιατί το πρόγραμμα θα εκτελείται επ΄ άπειρον

20. Ποιος είναι ο μέγιστος αριθμός κόμβων που δεν έχουν κόμβους-απογόνους σε ένα δυαδικό δέντρο με d επίπεδα;
◯ A. 2^d
◯ B. $2^d - 1$
◯ Γ. 2^{d-1}

21. Ποια είναι η πολυπλοκότητα χρόνου για την εισαγωγή ενός κόμβου σε ένα ταξινομημένο και πλήρως ισοζυγισμένο δυαδικό δέντρο με n κόμβους, ώστε αυτό να εξακολουθήσει να είναι ταξινομημένο;
◯ A. $O(n)$
◯ B. $O(n^2)$
◯ Γ. $O(logn)$

22. Η μνήμη που απαιτείται για τη φύλαξη σε μία λίστα ενός συνόλου ακεραίων σε σχέση με αυτή που απαιτείται για τη φύλαξη των ίδιων ακεραίων σε ένα πίνακα είναι ...
◯ A. ... λιγότερη
◯ B. ... περισσότερη
◯ Γ. ... ακριβώς ίση

23. Η γρήγορη μέθοδος ταξινόμησης (quicksort) ...
◯ A. ... είναι επαναληπτική
◯ B. ... είναι αναδρομική
◯ Γ. ... απαιτεί δυναμική δέσμευση μνήμης για να λειτουργήσει σωστά

24. Πόσες εξωτερικές/καθολικές μεταβλητές βρίσκονται συνολικά στα προγράμματα των σημειώσεων/διαφανειών του μαθήματος;
◯ A. 0
◯ B. 1
◯ Γ. 50

25. Το τμήμα που σπουδάζετε γίνεται φέτος ...
◯ A. ... 25 ετών
◯ B. ... 30 ετών
◯ Γ. ... 35 ετών

Θέμα 2 (30/100): Εφ΄ όσον η βαθμολογία στο **Θέμα 1** είναι τουλάχιστον **10/50**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Ένας αριθμός ονομάζεται *αυτομορφικός* (automorphic) όταν το τετράγωνό του έχει ως τελευταία ψηφία τον ίδιο τον αριθμό. Για παράδειγμα, ο αριθμός 25 είναι αυτομορφικός ($25^2 = 625$). Επίσης και ο αριθμός 9376 ($9376^2 = 87909376$). Γράψτε ένα πρόγραμμα C το οποίο να δέχεται στη γραμμή εντολής ένα θετικό ακέραιο N και να βρίσκει όλους τους αυτομορφικούς αριθμούς που είναι μικρότεροι του N. Σημειώστε ότι **δεν επιτρέπεται να χρησιμοποιήσετε πραγματικούς αριθμούς, συναρτήσεις της μαθηματικής βιβλιοθήκης και πίνακες** (εξαιρείται το όρισμα `argv[]` της `main()`). Μία ενδεικτική εκτέλεση:

```
$ ./automorphic 30000
1 5 6 25 76 376 625 9376
```

Θέμα 3 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1 και 2** είναι τουλάχιστον **40/80**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί.

Μία απλή επαναληπτική μέθοδος η οποία, κάτω από κάποιες προϋποθέσεις, είναι σε θέση να βρει μία ρίζα συνάρτησης (τιμή της ανεξάρτητης μεταβλητής που μηδενίζει τη συνάρτηση) είναι ο αλγόριθμος Newton-Raphson. Σύμφωνα με τη μέθοδο αυτή, αρχίζουμε από κάποια αυθαίρετη εκτίμηση μίας ρίζας της συνάρτησης, έστω x_0, και εφαρμόζοντας τον επαναληπτικό τύπο $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, υπολογίζουμε διαδοχικές προσεγγίσεις της ρίζας. Όταν δύο συνεχόμενες προσεγγίσεις διαφέρουν ελάχιστα (λιγότερο από κάποιο πολύ μικρό $\epsilon > 0$), σταματάμε και η τελευταία προσέγγιση που υπολογίστηκε θεωρούμε ότι είναι η ρίζα που ζητάμε. Η παράγωγος της συνάρτησης μπορεί να υπολογισθεί προσεγγιστικά από τον τύπο $f'(x_n) = \frac{f(x_n+\epsilon)-f(x_n)}{\epsilon}$.

Ορίστε το πρωτότυπο συνάρτησης C με όνομα `findaroot`, και στη συνέχεια υλοποιήστε την, η οποία να δέχεται σαν ορίσματα μία συνάρτηση f, με όρισμα `double` και επιστροφή `double`, μία αρχική εκτίμηση ρίζας της συνάρτησης x_0 και ένα μικρό θετικό αριθμό ϵ και να επιστρέφει, εφαρμόζοντας την προηγούμενη μέθοδο, μία ρίζα της συνάρτησης. Επίσης, υλοποιήστε και μία συνάρτηση `main` που να καλεί την `findaroot` με κατάλληλα ορίσματα (η αρχική εκτίμηση x_0 να δίνεται από την είσοδο, αλλά ως ϵ θεωρήστε το 10^{-7}). Ενδεικτικές εκτελέσεις (για τη συνάρτηση $f(x) = x^3 - 2x^2 - 11x + 12$)[1]:

```
$ ./newtraph
Give starting guess of root: 0.0
root = 1.000000000
$ ./newtraph
Give starting guess of root: 2.5
root = -3.000000000
$ ./newtraph
Give starting guess of root: 3.5
root = 4.000000000
```

[1] Προσοχή, η `findaroot` πρέπει να είναι σε θέση να λειτουργήσει για οποιαδήποτε συνάρτηση της δοθεί ως όρισμα και όχι μόνο για τη συγκεκριμένη του παραδείγματος.

Θέμα 4 (20/100): Εφ΄ όσον η συνολική βαθμολογία από τα **Θέματα 1, 2 και 3** είναι τουλάχιστον **80/100**, στον τελικό βαθμό θα συνυπολογισθεί και η αξιολόγηση της απάντησής σας στο πρόβλημα που ακολουθεί. Ο τελικός βαθμός δεν μπορεί να υπερβαίνει το **100/100**.

Σας δίνεται ο παρακάτω κώδικας C:

```
void e(int *c, int p1, int p2)
{  int tmp;
   tmp = c[p1] ; c[p1] = c[p2] ; c[p2] = tmp; }

void g(int *c, int k, int n)
{  int i, j;
   if (k == n) {
      for (j = 0 ; j < n ; j++) printf("%d ", c[j]);
      printf("\n"); }
   else {
      for (i = k ; i < n ; i++) {
         e(c, k, i); g(c, k + 1, n); e(c, k, i); } } }

void f(int n)
{  int i, *c;
   c = malloc(n * sizeof(int));
   for (i = 0 ; i < n ; i++) c[i] = i + 1;
   g(c, 0, n);
   free(c); }
```

1. Τι θα εκτυπωθεί στην έξοδο αν η συνάρτηση `f()` κληθεί ως `f(3)`;

2. Πόσες γραμμές θα εκτυπωθούν στην έξοδο αν κληθεί ως `f(10)`;

Αιτιολογήστε στοιχειωδώς τις απαντήσεις σας. Σε κάθε περίπτωση, θεωρήστε ότι η συνάρτηση `malloc()` που καλείται επιστρέφει με επιτυχία.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ
Εξετάσεις Β΄ Περιόδου 2020 (31/8/2020)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....

Α.Μ.: .....

Θέμα 1 (60/100): Επιλέξτε στις παρακάτω ερωτήσεις ή δηλώσεις τη σωστή απάντηση. Κάθε **σωστή απάντηση προσθέτει 2 μονάδες** στον τελικό βαθμό και κάθε **λάθος απάντηση αφαιρεί 1 μονάδα** από τον τελικό βαθμό.

1. Με ποιον τρόπο μπορεί να εκτελεσθεί ο κώδικας που περιέχεται σε ένα πηγαίο αρχείο C με όνομα `prog.c`;
   ◯ Α. Εκτελώντας στο τερματικό την εντολή `./prog.c`
   ◯ Β. Κάνοντας σε γραφικό περιβάλλον διπλό κλικ στο όνομα του πηγαίου αρχείου
   ◯ Γ. Καμία από τις άλλες απαντήσεις δεν είναι σωστή

2. Για να παραχθεί από ένα πηγαίο αρχείο C με όνομα `prog.c` το αντίστοιχο αντικειμενικό αρχείο `prog.o` με τον μεταγλωττιστή `gcc` σε ένα περιβάλλον τερματικού, αρκεί να κληθεί ο μεταγλωττιστής με όρισμα το όνομα του πηγαίου αρχείου και ...
   ◯ Α. ... με επιπλέον επιλογή την `-c`
   ◯ Β. ... με επιπλέον επιλογή την `-o`
   ◯ Γ. ... χωρίς καμία επιπλέον επιλογή

3. Ποια τιμή θα έχει η ακέραια μεταβλητή `x` μετά την εκτέλεση της εντολής `x = 3 << 2;`
   ◯ Α. 0
   ◯ Β. 1
   ◯ Γ. 12

4. Ποια τιμή θα έχει η ακέραια μεταβλητή `x` μετά την εκτέλεση της εντολής `x = 3 < 2;`
   ◯ Α. 0
   ◯ Β. 1
   ◯ Γ. 12

5. Ποια τιμή θα έχει η ακέραια μεταβλητή `x` μετά την εκτέλεση της εντολής `x = 5 | 2;`
   ◯ Α. 1
   ◯ Β. 7
   ◯ Γ. 25

6. Ποια τιμή θα έχει η ακέραια μεταβλητή `x` μετά την εκτέλεση της εντολής `x = 5 || 2;`
   ◯ Α. 1
   ◯ Β. 7
   ◯ Γ. 25

7. Σε ποιο χώρο φυλάσσεται η τιμή μίας μεταβλητής που έχει ορισθεί στο εσωτερικό μίας συνάρτησης;
   ◯ Α. Στη στοίβα, σε κάθε περίπτωση
   ◯ Β. Στη στοίβα, εκτός αν έχει ορισθεί η μεταβλητή ως `dynamic`, οπότε φυλάσσεται στο σωρό που φυλάσσονται και οι τιμές των μεταβλητών που έχουν ορισθεί δυναμικά
   ◯ Γ. Στη στοίβα, εκτός αν έχει ορισθεί η μεταβλητή ως `static`, οπότε φυλάσσεται στον εξωτερικό χώρο που φυλάσσονται και οι τιμές των καθολικών μεταβλητών

8. Μία συνάρτηση C είναι αναδρομική ...
   ◯ Α. ... όταν καλεί (άμεσα ή έμμεσα) τον εαυτό της
   ◯ Β. ... όταν στον ορισμό της έχει δηλωθεί ως `recursive`
   ◯ Γ. ... όταν διαχειρίζεται λίστες ή δυαδικά δέντρα

9. Πότε μπορούμε να βρούμε πιο γρήγορα τη δυαδική αναπαράσταση ενός θετικού ακεραίου αριθμού;
   ◯ Α. Όταν γνωρίζουμε την τιμή του στο δεκαδικό σύστημα αρίθμησης
   ◯ Β. Όταν γνωρίζουμε την τιμή του στο δεκαεξαδικό σύστημα αρίθμησης
   ◯ Γ. Όταν γνωρίζουμε την ανάλυση του αριθμού σε γινόμενο πρώτων παραγόντων, για τους οποίους έχουμε τη δυαδική αναπαράστασή τους

10. Πόσα bytes απαιτούνται για τη φύλαξη στη μνήμη της συμβολοσειράς `"mystring"`, υποθέτοντας ότι η κωδικοποίηση των χαρακτήρων της είναι κατά ASCII;
    - ◯ Α. 8
    - ◯ Β. 9
    - ◯ Γ. 10

11. Για την ακέραια μεταβλητή `i` της C, είναι αληθές ότι η τιμή της έκφρασης `(double) i * (double) i` δεν έχει διαφορά από την τιμή της έκφρασης `(double) (i * i)`;
    - ◯ Α. Ναι, ισχύει πάντοτε
    - ◯ Β. Όχι, ποτέ δεν ισχύει
    - ◯ Γ. Εξαρτάται από την τιμή της μεταβλητής `i` αν ισχύει ή όχι

12. Ο ορισμός της συνάρτησης `main` ως `int main(void)` σημαίνει ότι ...
    - ◯ Α. ... η συνάρτηση δεν πρόκειται να καλέσει άλλες συναρτήσεις
    - ◯ Β. ... δεν μας ενδιαφέρει το εκτελέσιμο πρόγραμμα που θα προκύψει να κληθεί με ορίσματα στη γραμμή εντολών
    - ◯ Γ. ... η συνάρτηση θα τερματίσει με επιτυχία

13. Ποια τιμή θα έχει η ακέραια μεταβλητή `x` μετά την εκτέλεση των εντολών `x=-5; x-=5;`
    - ◯ Α. 0
    - ◯ Β. -5
    - ◯ Γ. -10

14. Αν οι χαρακτήρες κωδικοποιούνται κατά ASCII, τότε στην C η τιμή της έκφρασης `'Q' - 'M' + 'm'` ισούται με την τιμή της έκφρασης ...
    - ◯ Α. ... `'q' -'m'`
    - ◯ Β. ... `'q'`
    - ◯ Γ. ... `'m' -'q'`

15. Ποιος είναι ο ελάχιστος αριθμός από bytes που απαιτούνται για να φυλαχθεί στη μνήμη ο δεκαεξαδικός αριθμός `0xB67FFE3AC017CD3`;
    - ◯ Α. 4
    - ◯ Β. 8
    - ◯ Γ. 16

16. Αν η μεταβλητή `x` είναι τύπου `int` και η μεταβλητή `p` είναι τύπου `int **`, τότε για τις εντολές `x = *(*p);` και `p = &(&x);` μπορούμε να πούμε ότι ...
    - ◯ Α. ... και οι δύο είναι συντακτικά σωστές και είναι λειτουργικά ισοδύναμες
    - ◯ Β. ... και οι δύο είναι συντακτικά σωστές, αλλά έχουν διαφορετική λειτουργικότητα
    - ◯ Γ. ... τουλάχιστον μία από τις δύο είναι συντακτικά λανθασμένη

17. Ποια από τις παρακάτω θα μπορούσε να είναι μία σωστή δήλωση πρωτοτύπου της συνάρτησης `square` που δέχεται έναν αριθμό `double` και υπολογίζει και επιστρέφει το τετράγωνό του;
    - ◯ Α. `void square(double);`
    - ◯ Β. `void square(double, double);`
    - ◯ Γ. `void square(double, double *);`

18. Ποια από τις παρακάτω **δεν** θα μπορούσε να είναι μία σωστή δήλωση πρωτοτύπου της συνάρτησης `square` που δέχεται έναν αριθμό `double` και υπολογίζει και επιστρέφει το τετράγωνό του;
    - ◯ Α. `double (*square)(double);`
    - ◯ Β. `double square(double);`
    - ◯ Γ. `(double *) square(double);`

19. Έστω ότι σε ένα πρόγραμμα C ορίζουμε τη δομή `struct mystr { ... }`. Ποιο από τα παρακάτω **δεν** θα μπορούσε να είναι ένα συντακτικά σωστό μέλος για τη δομή αυτή;
    - ◯ Α. `struct mystr s;`
    - ◯ Β. `struct mystr *ps;`
    - ◯ Γ. `struct mystr **pps;`

20. Πότε υπάρχει κίνδυνος να υπερχειλίσει η στοίβα εκτέλεσης ενός προγράμματος;
    - ◯ Α. Όταν δεσμεύουμε δυναμικά μεγάλους πίνακες
    - ◯ Β. Όταν διαχειριζόμαστε απλά συνδεδεμένες λίστες με αναδρομικό τρόπο
    - ◯ Γ. Όταν ορίζουμε μεγάλους εξωτερικούς/καθολικούς πίνακες

21. Ο καλύτερος αλγόριθμος που μπορούμε να γράψουμε για να αντιστρέψουμε τη σειρά των στοιχείων (πλήθους n) μίας απλά συνδεδεμένης λίστας έχει πολυπλοκότητα ...
◯ Α. ... $O(logn)$
◯ Β. ... $O(n)$
◯ Γ. ... $O(n^2)$

22. Ποια τιμή θα επιστραφεί από την παρακάτω συνάρτηση C, αν αυτή κληθεί ως `f(6);`

```
int f(int x)
{ return x ? x * f(x-1) : 1; }
```

◯ Α. 0
◯ Β. 21
◯ Γ. 720

23. Συμφωνείτε ότι η δυαδική αναζήτηση σε πίνακα απαιτεί την οργάνωση των στοιχείων του πίνακα σε ένα δυαδικό δέντρο;
◯ Α. Όχι, δεν συμφωνώ
◯ Β. Ναι, υπό την προϋπόθεση ότι το δέντρο είναι ταξινομημένο
◯ Γ. Ναι, υπό την προϋπόθεση ότι το δέντρο είναι ισοζυγισμένο

24. Σε ένα δυαδικό δέντρο, όλα τα μονοπάτια από τη ρίζα σε κόμβους-φύλλα έχουν το ίδιο μήκος n, θεωρώντας ότι η απόσταση ενός κόμβου από απόγονό του είναι μοναδιαία (έχει μήκος 1). Πόσοι είναι συνολικά όλοι οι κόμβοι του δέντρου;
◯ Α. $2 \cdot n - 1$
◯ Β. 2^n
◯ Γ. $2^{n+1} - 1$

25. Αν το πλήθος των ακεραίων με πρόσημο που μπορούν να φυλαχθούν σε 2 bytes ισούται με K και το πλήθος των ακεραίων χωρίς πρόσημο που μπορούν να φυλαχθούν σε 8 bytes ισούται με M, τότε ισχύει ...
◯ Α. ... $M = K^4$
◯ Β. ... $M = K^2$
◯ Γ. ... $M = 4^K$

26. Στο αρχείο επικεφαλίδας `math.h` περιέχεται, μεταξύ άλλων, και ...
◯ Α. ... η υλοποίηση της συνάρτησης `sqrt` σε πηγαία μορφή
◯ Β. ... η υλοποίηση της συνάρτησης `sqrt` σε γλώσσα assembly
◯ Γ. ... το πρωτότυπο `double sqrt(double);` της συνάρτησης `sqrt`

27. Ποια τιμή θα έχει η ακέραια μεταβλητή `x` μετά την εκτέλεση της εντολής `x = 12/5.0 + 12.0/5 + 12/5;`
◯ Α. 6
◯ Β. 7
◯ Γ. 8

28. Για να είναι ένας ακέραιος N μεγαλύτερος του 2 πρώτος, αρκεί να ...
◯ Α. ... μην διαιρείται με κανέναν από τους αριθμούς 2, 3, 5 ή 7
◯ Β. ... είναι περιττός και να μην έχει διαιρέτη μεγαλύτερο του 1 και μικρότερο του $\sqrt{N}$
◯ Γ. ... μην έχει διαιρέτη μεγαλύτερο του 1 και μικρότερο ή ίσο του $N/2$

29. Αν θέλαμε να ταξινομήσουμε ένα πίνακα με τη γρήγορη μέθοδο ταξινόμησης, πότε θα αναμέναμε να έχει η μέθοδος αυτή τη χειρότερη απόδοση;
◯ Α. Όταν τα στοιχεία του πίνακα είναι ακέραιοι αριθμοί σε εντελώς τυχαία σειρά
◯ Β. Όταν τα στοιχεία του πίνακα είναι αριθμοί κινητής υποδιαστολής σε εντελώς τυχαία σειρά
◯ Γ. Όταν τα στοιχεία του πίνακα είναι οποιουδήποτε τύπου, ήδη ταξινομημένα σε αύξουσα σειρά

30. Συμφωνείτε ότι είναι εφικτό να γράψουμε ένα πρόγραμμα C το οποίο να υπολογίζει με απόλυτη ακρίβεια τον γνωστό αριθμό $\pi = 3.14\ldots$;
◯ Α. Ναι, αλλά το πρόγραμμα θα έχει τουλάχιστον 1000 γραμμές κώδικα
◯ Β. Ναι, αλλά το πρόγραμμα θα πρέπει να εκτελεσθεί σε υπερυπολογιστή, σαν αυτούς που χρησιμοποιούν στην Google ή στη NASA
◯ Γ. Δεν συμφωνώ, γιατί κάτι τέτοιο δεν είναι εφικτό και ούτε πρόκειται ποτέ να γίνει

Θέμα 2 (40/100): Γράψτε ένα πρόγραμμα C το οποίο να βρίσκει τον **μικρότερο θετικό ακέραιο** x για τον οποίο υπάρχουν **θετικοί ακέραιοι** a, b, c, d, τέτοιοι ώστε να ισχύει $x^4 = a^4 + b^4 + c^4 + d^4$. Για παράδειγμα, ένας αριθμός με αυτή την ιδιότητα, όχι όμως ο μικρότερος δυνατός, είναι ο 651, αφού ισχύει ότι $651^4 = 240^4 + 340^4 + 430^4 + 599^4$. **Δεν επιτρέπεται να χρησιμοποιήσετε πραγματικούς αριθμούς, συναρτήσεις της μαθηματικής βιβλιοθήκης και πίνακες**.

Θέμα 3 (20/100): Θεωρήστε ότι το εκτελέσιμο πρόγραμμα που θα προκύψει από το παρακάτω πηγαίο πρόγραμμα C καλείται στη γραμμή εντολών με όρισμα ένα θετικό ακέραιο. Γράψτε μία εκφώνηση θέματος εξετάσεων για την οποία το πηγαίο αυτό πρόγραμμα θα ήταν μία σωστή απάντηση.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{   int n, f, b, p = 2, s = 5;
    if (argc < 2) return 1;
    n = atoi(argv[1]);
    if (n-- > 1) {
        p *= 3;
        while (n > 1) {
            f = 3; b = 1;
            while (f * f <= s && b) {
                if (!(s % f)) b = 0;
                f += 2; }
            if (b) {
                p *= s; n--; }
            s += 2; } }
    printf("%d\n", p);
    return 0; }
```

Θέμα 4 (20/100): Η συνάρτηση C που φαίνεται αριστερά είναι η υλοποίηση της δυαδικής αναζήτησης σε πίνακα συμβολοσειρών που υπάρχει στις σημειώσεις/διαφάνειες του μαθήματος. Δεξιά φαίνεται μία μικρή παραλλαγή της. Οι δύο συναρτήσεις διαφέρουν μόνο στη γραμμή `mid = ...`

```
int binsearch(char *w, int n, char **x)
{ int cond, low, high, mid;
  low = 0;
  high = n-1;
  while (low <= high) {
    mid = (low+high)/2;
    if ((cond = strcmp(w, x[mid])) < 0)
      high = mid-1;
    else if (cond > 0)
      low = mid+1;
    else return 1; }
  return 0; }
```

```
int binsearch(char *w, int n, char **x)
{ int cond, low, high, mid;
  low = 0;
  high = n-1;
  while (low <= high) {
    mid = low+(high-low)/2;
    if ((cond = strcmp(w, x[mid])) < 0)
      high = mid-1;
    else if (cond > 0)
      low = mid+1;
    else return 1; }
  return 0; }
```

Κάποιοι ισχυρίζονται ότι η συνάρτηση στα αριστερά έχει σφάλμα, το οποίο σε ορισμένες περιπτώσεις μπορεί να δώσει λάθος αποτελέσματα, και προτείνουν η συνάρτηση να γραφεί όπως φαίνεται δεξιά. Εξηγήστε γιατί είναι σωστός ο ισχυρισμός αυτός.

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....
SDI (sdiYYOONNN): .....

Εισαγωγή στον Προγραμματισμό - Ιούλιος 2024

Επί Πτυχίω Εξέταση

Διάρκεια: 120 λεπτά

Απαντήστε και στα 4 θέματα, τα οποία είναι βαθμολογικά ισοδύναμα. Τα προγράμματα που θα γράψετε πρέπει να είναι δομημένα, διατυπωμένα ευκρινώς εντός του διαθέσιμου χώρου και με επαρκή τεκμηρίωση ώστε να είναι κατανοητά.

Θέμα 1ο - ASCII Encoding (25 Μονάδες)

Γράψτε ένα πρόγραμμα το οποίο διαβάζει όλους τους χαρακτήρες που δίνονται από την πρότυπη είσοδο (stdin) και τυπώνει στην πρότυπη έξοδο (stdout) την δεκαεξαδική αναπαράστασή τους με μία εξαίρεση: οι χαρακτήρες νέας γραμμής (newline) πρέπει να τυπώνονται ως έχουν - χωρίς μετατροπή σε δεκαεξαδική μορφή. Παράδειγμα εκτέλεσης ακολουθεί:

```
$ cat ithaka.txt
As you set out for Ithaka
hope your road is a long one,
full of adventure, full of discovery.
$ ./ascii < ithaka.txt
417320796f7520736574206f757420666f7220497468616b61
686f706520796f757220726f6164206973732061206c6f6e67206f6e652c
66756c6c206f6620616476656e747572652c2066756c6c206f66206469736f766572792e
```

Θέμα 2ο - Στατιστικές (25 Μονάδες)

Γράψτε ένα πρόγραμμα το οποίο δέχεται ακεραίους ως ορίσματα από την γραμμή εντολών και τυπώνει στην πρότυπη έξοδο (stdout) τον μέσο όρο τους και την τυπική απόκλιση με 3 δεκαδικά ψηφία ακρίβεια. Υπενθυμίζεται ότι ο μέσος όρος μ για N όρους δίνεται από τον τύπο: $\mu = \frac{\sum x_i}{N}$ και η τυπική απόκλιση δίνεται από τον τύπο: $\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$. Παραδείγματα εκτέλεσης ακολουθούν:

```
$ ./stats 8 4 12
Mean: 8.000, Standard deviation: 3.266
$ ./stats 8 4 12 23 2 3 42 432 2 22 3 4 2 3 2 32 3 23
Mean: 34.556, Standard deviation: 97.112
```

Θέμα 3ο - Βέλτιστη Μοιρασιά Πίτσας (25 Μονάδες)

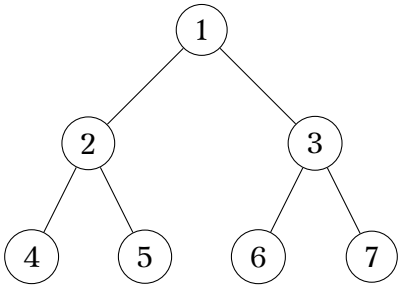
Παραγγείλατε πίτσες για να δείτε τον τελικό του Euro και διαπιστώσατε τελευταία στιγμή πως ήρθαν υπερβολικά πολλοί καλεσμένοι (ως συνήθως κάποιοι αυτοπροσκλήθηκαν). Παρόλα αυτά κάνατε την καρδιά σας πέτρα και αποφασίσατε να ταΐσετε όσους περισσότερους γίνεται. Και τι καλύτερος τρόπος για να γίνει αυτό παρά μέσα από ένα πρόγραμμα - περνάς ευχάριστα τον χρόνο σου όσο παίζουν οι διαφημίσεις.

Γράψτε ένα πρόγραμμα το οποίο διαβάζει από την πρότυπη είσοδο: (1) τον αριθμό από πίτσες που έχουμε, (2) τον αριθμό κομματιών κάθε πίτσας, (3) τον αριθμό των ατόμων που έχουμε, (4) τον αριθμό των κομματιών που επιθυμεί το κάθε άτομο και τυπώνει στην πρότυπη έξοδο τον μέγιστο αριθμό ατόμων που μπορούμε να ικανοποιήσουμε **πλήρως** (αν κάποιο άτομο επιθυμεί 5 κομμάτια, πρέπει να φάει 5, αν απλά φάει 1 δεν αρκεί). Περιγράψτε την χρονική και χωρική πολυπλοκότητα του αλγορίθμου σας και εξηγήστε αν είναι η βέλτιστη (8/25 της βαθμολογίας). Παράδειγμα εκτέλεσης ακολουθεί:

```
$ ./pizza
Number of pizzas available: 3
Enter the number of slices for each pizza: 8 10 5
Number of people: 4
Number of slices desired by each person: 6 9 7 5
Max number of people that can be satisfied: 3
```

Θέμα 4ο - Reverse Inorder Traversal (25 Μονάδες)

Γράψτε μια συνάρτηση `reverse_inorder` η οποία παίρνει ως όρισμα ένα δέντρο ακεραίων τύπου `Tree` και τυπώνει τους αριθμούς των κόμβων σε σειρά reverse inorder traversal, δηλαδή όπως βλέπουμε τους αριθμούς από δεξιά προς τα αριστερά (πρώτο το δεξί παιδί, μετά ο γονιός και στην συνέχεια το αριστερό). Για παράδειγμα, για το ακόλουθο δέντρο:



περιμένουμε να εκτυπωθεί η ακολουθία: 7 3 6 1 5 2 4. Ποια είναι η χρονική και η χωρική πολυπλοκότητα του αλγορίθμου σας (8/25 της βαθμολογίας); Ο τύπος `Tree` δίνεται παρακάτω:

```
typedef struct node {
  int value;
  struct node * left;
  struct node * right;
} * Tree;
```

Βοηθήματα

ASCII Table

Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec
NUL	0	NAK	21	*	42	?	63	T	84	i	105
SOH	1	SYN	22	+	43	@	64	U	85	j	106
STX	2	ETB	23	,	44	A	65	V	86	k	107
ETX	3	CAN	24	-	45	B	66	W	87	l	108
EOT	4	EM	25	.	46	C	67	X	88	m	109
ENQ	5	SUB	26	/	47	D	68	Y	89	n	110
ACK	6	ESC	27	0	48	E	69	Z	90	o	111
BEL	7	FS	28	1	49	F	70	[	91	p	112
BS	8	GS	29	2	50	G	71	\	92	q	113
HT	9	RS	30	3	51	H	72	]	93	r	114
LF	10	US	31	4	52	I	73	^	94	s	115
VT	11	Space	32	5	53	J	74	_	95	t	116
FF	12	!	33	6	54	K	75	`	96	u	117
CR	13	"	34	7	55	L	76	a	97	v	118
SO	14	#	35	8	56	M	77	b	98	w	119
SI	15	$	36	9	57	N	78	c	99	x	120
DLE	16	%	37	:	58	O	79	d	100	y	121
DC1	17	&	38	;	59	P	80	e	101	z	122
DC2	18	'	39	<	60	Q	81	f	102	{	123
DC3	19	(	40	=	61	R	82	g	103	|	124
DC4	20	)	41	>	62	S	83	h	104	}	125

Πρόχειρο

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: .....
SDI (sdiYYOONNN): .....

Εισαγωγή στον Προγραμματισμό - Σεπτέμβριος 2024

Επαναληπτική Εξεταστική

Διάρκεια: 135 λεπτά / Σύνολο: 100 Μονάδες (+5 bonus)

Τα προγράμματα C που θα γράψετε πρέπει να είναι δομημένα, διατυπωμένα ευκρινώς εντός του διαθέσιμου χώρου και με επαρκή τεκμηρίωση ώστε να είναι κατανοητά.

1. About [5 Μονάδες]

Γράψτε μια συνάρτηση `about` η οποία τυπώνει 3 γραμμές στην πρότυπη έξοδο (stdout): 1) το όνομά σας με λατινικούς χαρακτήρες στην 1η, 2) το sdi σας στην 2η και 3) `I'm 100% "ready"!` στην 3η. Παράδειγμα εξόδου από την εκτέλεση της συνάρτησης `about`:

```
Michael Jordan
sdi2300999
I'm 100% "ready"!
```

Απάντηση:

2. Η συνάρτηση what (10 Μονάδες)

```
int what(int *array, int x, int y) {
  int mid, low = 0, high = y - 1;
  while (low <= high) {
    mid = low + (high - low) / 2;
    printf("%d\n", mid);
    if (array[mid] == x) return 1;
    else if (array[mid] < x) low = mid + 1;
    else high = mid - 1;
  }
  return 0;
}
```

Τι κάνει η συνάρτηση what (μέχρι 10 λέξεις εξήγηση); Τι θα τυπώσει αν κληθεί με ορίσματα `{57, 98, 105, 241}`, `36`, `4`;
Απάντηση:

3. Εύρεση Μηδενός σε Πίνακα [15 Μονάδες]

Γράψτε μία συνάρτηση `find_zero` η οποία λαμβάνει ως όρισμα έναν πίνακα από τιμές double ταξινομημένες σε αύξουσα σειρά και επιστρέφει την θέση (index) όπου υπάρχει το 0 ή -1 αν δεν το βρει. Η συνάρτηση μπορεί να έχει οποιαδήποτε διεπαφή (ορίσματα / τύπο επιστροφής) επιθυμείτε. Για παράδειγμα αν δοθεί ο πίνακας `{-8.1, -2.3, 0.0, 1.9}` θέλουμε να επιστραφεί η τιμή 2. Αντίστοιχα, αν δοθεί ο πίνακας `{1.0}` θέλουμε να επιστραφεί -1. Τι χρονική και χωρική πολυπλοκότητα έχει ο αλγόριθμός σας (5/15 της βαθμολογίας);
Απάντηση:

4. Μετρητής Λέξεων [25 Μονάδες]

Γράψτε ένα πρόγραμμα `wordcount` το οποίο διαβάζει ένα κείμενο από την πρότυπη είσοδο (stdin) και τυπώνει στην πρότυπη έξοδο (stdout) τον αριθμό των λέξεων που διάβασε καθώς και το μέσο μήκος λέξης του κειμένου με ακρίβεια δύο δεκαδικών ψηφίων. Ως *λέξη*, ορίζουμε οποιαδήποτε ακολουθία συνεχόμενων χαρακτήρων A-Z, a-z και 0-9 (μετράμε μόνο λέξεις με λατινικούς χαρακτήρες) και ως *μήκος* τον αριθμό των συνεχόμενων χαρακτήρων. Για παράδειγμα, το κείμενο "I'm a good person" έχει σύνολο 5 λέξεις και μέσο μήκος (1 + 1 + 1 + 4 + 6)/5 = 2.6 χαρακτήρες. Παραδείγματα εκτέλεσης ακολουθούν:

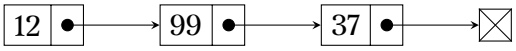
```
$ echo "I'm a good person" | ./wordcount
Total words: 5
Average word length: 2.60
```

```
$ cat quote.txt
"Success is not final, failure is not fatal: It is the courage to continue that counts."
$ ./wordcount < quote.txt
Total words: 16
Average word length: 4.25
```

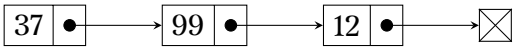
Απάντηση:

5. Αντιστροφή Λίστας [25 Μονάδες]

Γράψτε μια συνάρτηση `reverse_list` η οποία παίρνει ως όρισμα μια λίστα ακεραίων τύπου `List` και επιστρέφει μια νέα λίστα με τους κόμβους της σε αντίστροφη σειρά σε σχέση με την αρχική. Για παράδειγμα, αν δοθεί η ακόλουθη λίστα:



περιμένουμε να επιστραφεί η ανεστραμμένη χωρίς παρενέργειες (side-effects) στην αρχική:



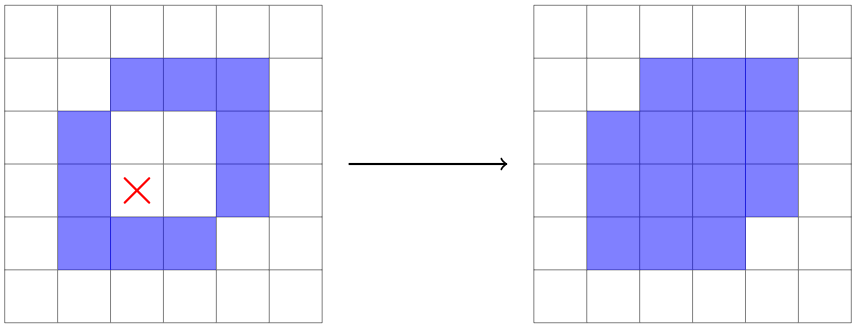
Ποια είναι η χρονική και η χωρική πολυπλοκότητα του αλγορίθμου σας (8/25 της βαθμολογίας); Ο τύπος `List` δίνεται παρακάτω:

```
typedef struct node {
  int value;
  struct node * next;
} * List;
```

Απάντηση:

6. Γεμίζοντας με Χρώμα [25 Μονάδες]

Αν έχετε χρησιμοποιήσει κάποιο πρόγραμμα ζωγραφικής στον υπολογιστή (π.χ., Paint), ίσως έχετε κάνει χρήση του "κουβά", του εργαλείου δηλαδή που μας βοηθάει να γεμίζουμε γρήγορα κλειστές επιφάνειες χωρίς εμπόδια. Το Σχήμα 1 δείχνει ένα παράδειγμα:



Σχήμα 1: Οπτικοποίηση της διαδικασίας γεμίσματος με χρώμα. Στο σχήμα στα αριστερά επιλέξαμε την θέση (3, 2) - 3η γραμμή, 2η στήλη - για να την γεμίσουμε με χρώμα. Στο σχήμα στα δεξιά παρατηρούμε το αποτέλεσμα του γεμίσματός μας.

Για τις ανάγκες της εφαρμογής μας κάνουμε τις εξής παραδοχές: (1) έχουμε μόνο δύο χρώματα που αναπαρίστανται με 0 και 1, (2) το πρόγραμμά μας γεμίζει την εικόνα μόνο με το χρώμα 1, (3) οι εικόνες που χειριζόμαστε είναι τετραγωνικές και (4) το χρώμα μπορεί μόνο να γεμίζει επιφάνειες που επικοινωνούν ελεύθερα με το αρχικό σημείο κάνοντας μόνο κινήσεις πάνω, κάτω, αριστερά, δεξιά (όχι διαγώνια). Εάν ένα κελί περιέχει ήδη το χρώμα 1, το κελί δεν θεωρείται ελεύθερο και δρα ως "τοίχος" για το γέμισμά μας.

Γράψτε ένα πρόγραμμα το οποίο παίρνει ως όρισμα το όνομα του αρχείου που περιέχει την εικόνα και το αρχικό σημείο για γέμισμα και τυπώνει στην έξοδο την τελική εικόνα. Η πρώτη σειρά του αρχείου περιέχει την διάσταση της εικόνας, η δεύτερη σειρά τις συντεταγμένες του αρχικού σημείου και τέλος ακολουθεί η δισδιάστατη εικόνα. Παράδειγμα εκτέλεσης ακολουθεί:

```
$ cat picture.txt
6
3 2
000000
001110
010010
010010
011100
000000
$ ./fill picture.txt
000000
001110
011110
011110
011100
000000
```

Απάντηση:

Πρόχειρο