# Ε.Α.Π./ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

**4η ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ**

**ΑΚΑΔΗΜΑΪΚΟΥ ΕΤΟΥΣ 2022-2023**

**3ος Τόμος**

## ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

**18/3/2023**

**Ημερομηνία παράδοσης εργασίας: Κυριακή 30/04/2023**

**Καταληκτική ημερομηνία παραλαβής: Τετάρτη**[[1]](#footnote-1) **03/05/2023**

**Ημερομηνία ανάρτησης ενδεικτικών λύσεων: Σάββατο 06/05/2023**

**Καταληκτική ημερομηνία αποστολής σχολίων στον φοιτητή: Πέμπτη 18/05/2022**

**ΥΠΟΕΡΓΑΣΙΑ 1.** (βαθμοί 25)

Πίνακες δύο διαστάσεων

**ΥΠΟΕΡΓΑΣΙΑ 2.** (βαθμοί 25)

Στοίβες

**ΥΠΟΕΡΓΑΣΙΑ 3.** (βαθμοί 30)

Διαπεράσεις δυαδικών δένδρων και δένδρα-σωροί

**ΥΠΟΕΡΓΑΣΙΑ 4.** (βαθμοί 20)

Συνδεδεμένες λίστες, Πίνακες δομών

**ΣΥΝΟΛΟ** **(βαθμοί 100)**

**ΥΠΟΕΡΓΑΣΙΑ 1. (βαθμοί 25)**

Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C που θα κάνει τις ακόλουθες ενέργειες:

**A)** Θα ορίζει έναν τετραγωνικό πίνακα “*pinakas*”, διαστάσεων N X N (ο ακέραιος Ν θα ορίζεται ως συμβολική σταθερά στην αρχή του προγράμματος π.χ. N=4). Θα διαβάζει από τον χρήστη NxN σε πλήθος θετικούς ακεραίους διαφορετικούς μεταξύ τους, οι οποίοι θα εισάγονται ανά γραμμή στον πίνακα “*pinakas”.* (Nα γίνει χρήση αμυντικού προγραμματισμού ώστε να εξασφαλιστεί ότι οι αριθμοί θα είναι διαφορετικοί μεταξύ τους και θετικοί).

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 12 | 6 | 8 |
| 17 | 25 | 199 | 78 |
| 100 | 133 | 121 | 144 |
| 55 | 67 | 88 | 44 |

**Πίνακας 1: Παράδειγμα πίνακα 4 x 4**

**B)** Θα εμφανίζει τα στοιχεία του πίνακα κατά γραμμές.

**Γ)** Θα υπολογίζει τον ανάστροφο πίνακα “*transpose\_pin*” και θα εμφανίζει τα στοιχεία του. Ο ανάστροφος ενός πίνακα προκύπτει με εναλλαγή μεταξύ γραμμών και στηλών του αρχικού πίνακα, δηλαδή το στοιχείο της i-οστής γραμμής και j-οστής στήλης του αρχικού πίνακα τοποθετείται στην j-οστή γραμμή και i-οστή στήλη του ανάστροφου.

**Δ)** Θα υπολογίζει το άθροισμα των στοιχείων της κύριας διαγώνιου του πίνακα “*pinakas*”, (δηλαδή των στοιχείων στα οποία ο αριθμός γραμμής ισούται με τον αριθμό στήλης) καθώς και του πίνακα “*transpose\_pin*”, τα οποία θα εμφανίζει. Τι παρατηρείτε για τα δύο αυτά αθροίσματα;

**Ε)** Διατρέχοντας τον πίνακα “pinakas”, θα εντοπίζει στα στοιχεία του τους πρώτους αριθμούς και θα τους εισάγει σε νέο μονοδιάστατο πίνακα “prime”, τον οποίον και θα εμφανίζει (εάν δεν υπάρχει κανένας πρώτος αριθμός θα εμφανίζει κατάλληλο μήνυμα). Διευκρινίζεται ότι ένας θετικός ακέραιος είναι πρώτος όταν διαιρείται μόνο από τον εαυτό του και την μονάδα. Η μονάδα δεν θεωρείται πρώτος αριθμός.

**ΣΤ)** Διατρέχοντας τον πίνακα “*prime*” θα υπολογίζει: (i) τις απόλυτες τιμές των διαφορών μεταξύ όλων των στοιχείων του και (ii) την μέγιστη τιμή αυτών των διαφορών την οποία και θα εμφανίζει με κατάλληλο μήνυμα.

Ακολουθεί ενδεικτικό παράδειγμα εκτέλεσης:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

**ΥΠΟΕΡΓΑΣΙΑ 2** **(βαθμοί 25)**

Τα προγράμματα περιήγησης ιστού διατηρούν τις διευθύνσεις που επισκεφθήκαμε στο ιστορικό τους, ώστε να είναι πιο εύκολη η επιστροφή σε αυτές που έχουμε επισκεφθεί. Γι’ αυτόν τον λόγο διαθέτουν το πλήκτρο «back» το οποίο μας μεταφέρει στην προηγούμενη σελίδα που επισκεφθήκαμε και το πλήκτρο «forward» το οποίο μας μεταφέρει στην σελίδα από την οποία επιστρέψαμε με τη χρήση του «back». Κάθε φορά που εισάγουμε μία διεύθυνση, αυτή καταχωρίζεται στο ιστορικό και απενεργοποιείται το πλήκτρο «forward».

Για την αποθήκευση του ιστορικού χρησιμοποιείται μια στοίβα παλαιότερων διευθύνσεων s1 και μία ακόμα βοηθητική στοίβα s2. Κάθε φορά που επισκεπτόμαστε μια νέα σελίδα, αυτή προστίθεται («push») στην κορυφή της στοίβας s1 και ό,τι και να περιέχει η στοίβα s2 διαγράφεται. Κάθε φορά που επιστρέφουμε (με το πλήκτρο «back»), η τελευταία σελίδα αφαιρείται (pop) από τη στοίβα s1 και η διεύθυνση αυτής προστίθεται (push) στη δεύτερη στοίβα s2. Αν πατήσουμε το πλήκτρο «back», όταν η στοίβα s1 είναι κενή, δε συμβαίνει κάτι. Κάθε φορά που πατάμε το πλήκτρο «forward» η κορυφή της στοίβας s2 αφαιρείται (pop) και προστίθεται (push) στη στοίβα s1. Αν πατήσουμε το πλήκτρο «forward» όταν η στοίβα s2 είναι κενή, δεν συμβαίνει κάτι.

Για παράδειγμα, έστω ότι έχουμε επισκεφθεί με τη σειρά τις σελίδες:

[google.com](http://google.com/) -> eap.gr -> study.eap.gr -> open.eap.gr

Αν ο χρήστης πατήσει 2 φορές «back», τότε θα μεταβεί στη σελίδα eap.gr.

Στο στιγμιότυπο αυτό, τα περιεχόμενα της στoίβας s1 είναι τα ακόλουθα:

[google.com](http://google.com/) -> eap.gr

Ενώ τα περιεχόμενα της στοίβας s2 είναι τα ακόλουθα:

open.eap.gr -> study.eap.gr

Αν στη συνέχεια πατήσει το πλήκτρο «forward», θα μεταβεί στη σελίδα study.eap.gr, και αν εισαγάγει μια ακόμα διεύθυνση (π.χ. [gmail.com](http://gmail.com/)), τότε αυτή θα τοποθετηθεί στην κορυφή της s1, οπότε τα νέα περιεχόμενα της στοίβας s1 θα είναι τα ακόλουθα:

[google.com](http://google.com/) -> eap.gr -> study.eap.gr -> [gmail.com](http://gmail.com/)

Εδώ η στοίβα s2 θα αδειάσει (θα είναι κενή), αφού όπως προαναφέρθηκε κάθε φορά που εισάγουμε μία διεύθυνση, αυτή καταχωρίζεται στο ιστορικό και απενεργοποιείται το πλήκτρο «forward».

**Ζητείται να αναπτύξετε πρόγραμμα C** που θα εμφανίζει ένα μενού και θα ζητάει από τον χρήστη την επιλογή μιας από τις ακόλουθες: Εισαγωγή, Πίσω (Back), Εμφάνιση, Εμπρός (Forward), Έξοδος.

* Αν ο χρήστης επιλέξει «Εισαγωγή» θα ζητηθεί μια διεύθυνση και θα τοποθετηθεί (push) στη στοίβα s1. Στην περίπτωση αυτή, αν η στοίβα επόμενων διευθύνσεων (s2) έχει περιεχόμενα, αυτά θα διαγραφούν και η στοίβα s2 θα μείνει κενή.
* Αν ο χρήστης επιλέξει «Πίσω», η τρέχουσα διεύθυνση θα διαγράφεται (θα γίνεται pop) από τη στοίβα s1, θα προστίθεται (θα γίνεται push) στην στοίβα s2 και θα εμφανίζεται στην οθόνη η κορυφή της στοίβας s1. Αν δεν υπήρχε διεύθυνση στην οποία μπορούμε να επιστρέψουμε (η στοίβα s1 είναι άδεια) να εμφανίζεται σχετικό μήνυμα στην οθόνη.
* Αν ο χρήστης επιλέξει «Εμφάνιση περιεχομένων στοίβας διευθύνσεων s1», το πρόγραμμα θα εμφανίζει τις διευθύνσεις που έχει επισκεφτεί ο χρήστης (περιεχόμενα στοίβας διευθύνσεων s1) με αντίστροφη σειρά (πρώτα η πιο πρόσφατη, τελευταία η πιο παλαιά).
* Αν ο χρήστης επιλέξει «Εμπρός», το πρόγραμμα θα αφαιρεί από τη στοίβα s2 (pop) την τρέχουσα διεύθυνση, θα την εισάγει (push) στη στοίβα s1 και θα την εμφανίζει στην οθόνη. Αν δεν υπήρχε «επόμενη» διεύθυνση (η στοίβα s2 είναι άδεια) θα εμφανίζεται σχετικό μήνυμα.

Το πρόγραμμα θα τερματίζει όταν ο χρήστης επιλέξει «Έξοδος».

Υλοποιήστε τις ακόλουθες συναρτήσεις σύμφωνα με το σχετικό υπόδειγμα ypoergasia\_2\_code\_template.c.

void push(stack \*sp, char address2[N]){ /\* ΚΩΔΙΚΑΣ ΣΥΝΑΡΤΗΣΗΣ push \*/}

int pop (stack \*sp, char address3[N]){ /\* ΚΩΔΙΚΑΣ ΣΥΝΑΡΤΗΣΗΣ pop \*/ }

void empty\_stack (stack \*sp){ /\* ΚΩΔΙΚΑΣ ΣΥΝΑΡΤΗΣΗΣ empty\_stack που αφαιρεί όλα τα περιεχόμενα της στοίβας \*/ }

void print\_stack (stack sp){ /\* ΚΩΔΙΚΑΣ ΣΥΝΑΡΤΗΣΗΣ print\_stack που εμφανίζει όλα τα περιεχόμενα της στοίβας\*/ }

Η συνάρτηση pop θα επιστρέφει 0 αν η λειτουργία pop πραγματοποιήθηκε με επιτυχία, διαφορετικά (αν η στοίβα ήταν άδεια) επιστρέφει κάποια άλλη τιμή. Στη συνέχεια περιγράφεται ένα παράδειγμα εκτέλεσης:

**Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα**

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

**ΥΠΟΕΡΓΑΣΙΑ 3** **(βαθμοί 30)**

Ο πίνακας που ακολουθεί περιέχει ακέραιους αριθμούς (που συμβολίζονται με τα γράμματα του αγγλικού αλφαβήτου A έως J) και αποτελεί τη συνεχόμενη αναπαράσταση Δυαδικού Δένδρου Αναζήτησης, στους κόμβους του οποίου περιέχονται ως πληροφορίες θετικοί ακέραιοι αριθμοί διαφορετικοί μεταξύ τους. Η τιμή -1 σε κάποια θέση του πίνακα υποδηλώνει ότι στην αντίστοιχη θέση του δένδρου δεν υπάρχει κόμβος.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| **A** | **B** | **C** | **D** | **-1** | **E** | **F** | **-1** | **G** | **-1** | **-1** | **H** | **I** | **-1** | **-1** | **-1** | **-1** | **J** |

Πριν σχεδιάσετε το δένδρο που αντιστοιχεί σε αυτή τη συνεχόμενη αναπαράσταση, απαντήστε στις παρακάτω ερωτήσεις, αιτιολογώντας τις απαντήσεις σας με βάση τη θέση των στοιχείων στον πίνακα:

i) Γιατί το δένδρο αυτό δεν μπορεί να είναι δένδρο-σωρός;

ii) Ποιο είναι το μικρότερο και ποιο το μεγαλύτερο στοιχείο του δένδρου;

iii) Ποιο είναι το ύψος του δένδρου;

Στη συνέχεια σχεδιάστε το δένδρο και απαντήστε στα ακόλουθα ερωτήματα:

iv) Δώστε: α) την προδιατεταγμένη διαπέραση, β) την ενδοδιατεταγμένη διαπέραση και γ) τη μεταδιατεταγμένη διαπέραση του δένδρου. Επισημαίνεται ότι η ενδοδιατεταγμένη διαπέραση μας δίνει την ολική διάταξη των στοιχείων μεταξύ τους.

v) Ποια σχέση (μεγαλύτερο ή μικρότερο) υπάρχει μεταξύ των στοιχείων B, C και G ανά δύο; Αιτιολογήστε την απάντησή σας.

vi) Ποια είναι η μορφή του δένδρου μετά τη διαγραφή του κόμβου C; Δώστε δύο εναλλακτικές λύσεις με βάση τους δύο διαφορετικούς τρόπους διαγραφής κόμβου σε δυαδικό δένδρο αναζήτησης.

vii) Δημιουργήστε ένα δένδρο-σωρό ελαχίστων εισάγοντας σε αυτό έναν προς έναν τους αριθμούς με τη σειρά που προκύπτουν από την προδιατεταγμένη διαπέραση του δένδρου του ερωτήματος iv. Δώστε μόνο την τελική μορφή του δένδρου.

**ΥΠΟΕΡΓΑΣΙΑ 4** **(βαθμοί 20)**

Αντικείμενο της συγκεκριμένης άσκησης είναι η διερεύνηση των δυνατοτήτων της C για την υλοποίηση ψηφοφοριών με μία παραλλαγή της μεθόδου **Borda Count[[2]](#footnote-2)**. Η προτεινόμενη μέθοδος κατατάσσει υποψηφίους κατά σειρά προτίμησης, βασιζόμενη σε ένα σύνολο ψηφοφοριών. Σε κάθε ψηφοφορία συμμετέχουν όλοι οι υποψήφιοι, και με βάση την κατάταξή τους κάθε υποψήφιος παίρνει ένα πλήθος βαθμών ίσο με τον αριθμό υποψηφίων που κατατάσσονται χαμηλότερα από αυτόν αυξημένο κατά ένα. Για παράδειγμα αν υπάρχουν *n* υποψήφιοι τότε, από μία ψηφοφορία o πρώτος υποψήφιος θα λάβει n  βαθμούς, ο δεύτερος n-1, και ούτω καθεξής, μέχρι τον τελευταίο υποψήφιο που θα λάβει 1 βαθμό. Στη συνέχεια οι βαθμοί κάθε υποψηφίου πολλαπλασιάζονται με το πλήθος των ψηφοφόρων στη συγκεκριμένη ψηφοφορία, οι βαθμοί από όλες τις ψηφοφορίες αθροίζονται, και ο κάθε υποψήφιος καταλήγει με ένα σύνολο βαθμών. Νικητής είναι ο υποψήφιος με το μεγαλύτερο σύνολο βαθμών.

Ας θεωρήσουμε ότι έχουμε τέσσερεις υποψηφίους, A,  B, C, D και ότι έγιναν 5 ψηφοφορίες όπου στη 1η ψήφισαν 14 ψηφοφόροι, στη 2η 4, στην 3η 7, στην 4η 12 και στην 5η 7.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Πλήθος Ψηφοφόρων**  **σε κάθε ψηφοφορία** | **14** | **4** | **7** | **12** | **7** |
| 1η επιλογή | D | D | C | A | A |
| 2η επιλογή | C | B | D | C | B |
| 3η επιλογή | A | C | B | D | D |
| 4η επιλογή | B | A | A | B | C |

Εφόσον υπάρχουν τέσσερεις υποψήφιοι η πρώτη θέση θα παίρνει 4 βαθμούς, η δεύτερη 3 βαθμούς, η τρίτη 2 βαθμούς και η τελευταία 1 βαθμό. Οπότε προκύπτει ο εξής πίνακας συγκομιδής βαθμολογιών όταν πολλαπλασιαστούν και με το πλήθος της εκάστοτε ψηφοφορίας:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Πλήθος Ψηφοφόρων** | **14** | **4** | **7** | **12** | **7** |
| 1η επιλογή (4 βαθμοί) | D (14\*4=56) | D (4\*4=16) | C (7 \*4=28) | A (12\*4=48) | A (7\*4=28) |
| 2η επιλογή (3 βαθμοί) | C (14\*3=42) | B (4\*3=12) | D (7\*3=21) | C (12\*3=36) | B (7\*3=21) |
| 3η επιλογή (2 βαθμοί) | A (14\*2=28) | C (4\*2=8) | B (7\*2=14) | D (12\*2=24) | D (7\*2=14) |
| 4η επιλογή (1 βαθμός) | B (14\*1=14) | A (4\*1=4) | A (7\*1=7) | B (12\*1=12) | C (7\*1=7) |

Επομένως ο συνολικός βαθμός κάθε υποψήφιου είναι:

Α:28+4+7+48+28=115

B:14+12+14+12+21=73

C:42+8+28+36+7=121

D:56+16+21+24+14=131 (Νικητής)

Καλείστε να υλοποιήσετε την παραλλαγή αυτή της τεχνικής Borda Count σε πρόγραμμα C. Για τη διαδικασία της υλοποίησης θεωρήστε ότι έχουμε Ν υποψήφιους και Μ ψηφοφορίες και ότι οι υποψήφιοι έχουν μοναδικά ονόματα πλήθους το πολύ L χαρακτήρων (οι τιμές Ν, L και M δηλώνονται ως συμβολικές σταθερές).

Κάθε υποψήφιος θα αποθηκεύεται σε μία δομή (struct) με δύο πεδία, το όνομα του υποψηφίου και το σύνολο των βαθμών του μετά την εφαρμογή του αλγορίθμου σε όλες τις ψηφοφορίες:

typedef struct candidate

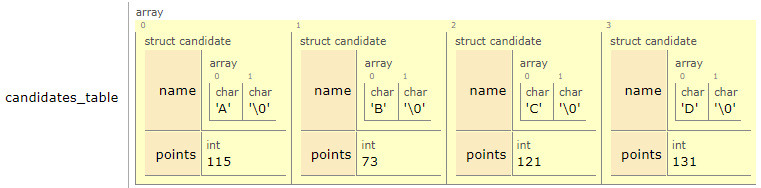
{

char name[L+1];

int points;

} candidate;

Το σύνολο των υποψηφίων θα αποθηκεύεται στον πίνακα δομών *candidates\_table*, που θα περιέχει τα ονόματα όλων των υποψηφίων, καθώς και το άθροισμα της βαθμολογίας τους μετά από κάθε μια ψηφοφορία, και δηλώνεται ως εξής: candidate candidates\_table[N];



Δημιουργήστε ένα πρόγραμμα σε C που θα διαβάζει το πλήθος των ψηφοφόρων για κάθε ψηφοφορία, καθώς και τα ονόματα των υποψηφίων με βάση τη σειρά κατάταξης τους σε κάθε ψηφοφορία, και θα εφαρμόζει τον αλγόριθμο Borda Count. Στο τέλος το πρόγραμμα θα εμφανίζει τον νικητή και το σύνολο βαθμών του. Πιο συγκεκριμένα, ζητούνται τα εξής:

**Α.** Να ορίσετε συνάρτηση με όνομα void Entercandidates(candidate candidates\_table[], int count) που θα ζητά από τον χρήστη τα ονόματα των υποψηφίων, και θα τα αποθηκεύει στο πεδίο *name* του κάθε κελιού του πίνακαcandidates\_table[] (δείτε το πιο πάνω σχήμα). Το πλήθος των υποψηφίων δίνεται από την παράμετρο *count*. Επίσης, πριν από την εισαγωγή ονομάτων στον πίνακα candidates\_table[], η συνάρτηση θα αρχικοποιεί σε μηδέν, τους βαθμούς του κάθε υποψηφίου και κατά την εισαγωγή θα εφαρμόζει αμυντικό προγραμματισμό ώστε να εξασφαλίζει ότι τα ονόματα των υποψηφίων είναι μοναδικά.

**Β.** Να ορίσετε συνάρτηση με όνομα void Entervotes(candidates\_table[], int count) η οποία θα δέχεται ως παράμετρο τον πίνακα candidates\_table[]με τα ονόματα των υποψηφίων όπως αυτά εισήχθησαν από την προηγούμενη συνάρτηση Entercandidates*.* Το πλήθος των υποψηφίων δίνεται από την παράμετρο *count*. Στη συνάρτηση αυτή, θα εισάγονται από τον χρήστη, το πλήθος των ψηφοφόρων που συμμετείχαν στην ψηφοφορία, και τα ονόματα των υποψηφίων με βάση τη σειρά κατάταξής τους στην ψηφοφορία. Τα ονόματα των υποψηφίωνθα εισάγονται σε μια απλά συνδεδεμένη λίστα με βάση τη σειρά κατάταξής τους (όπως φαίνεται στην παρακάτω εικόνα)**.** Κάθε κόμβος της συνδεδεμένης λίστας θα έχει την πιο κάτω δομή *node*, η οποία θα περιέχει το όνομα ενός υποψηφίου, και έναν δείκτη στον επόμενο κόμβο τύπου *node*.

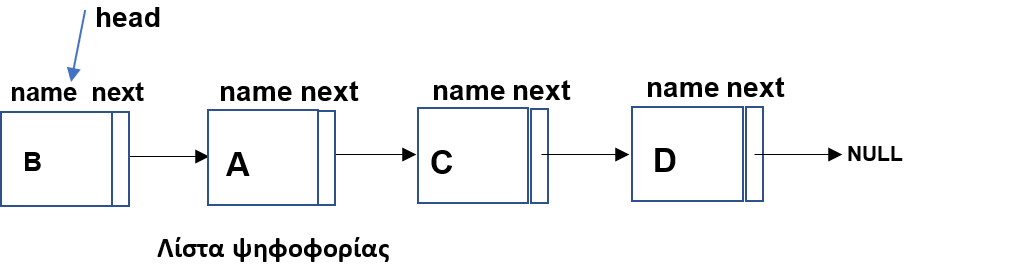
typedef struct node /\* Κόμβος λίστας ψηφοφορίας \*/

{

char name[L+1]; /\* Όνομα υποψηφίου \*/

struct node \* next; /\* Επόμενος στη λίστα ψηφοφορίας \*/

} node;



Στην παραπάνω συνδεδεμένη λίστα, καταδεικνύεται η σειρά κατάταξης των υποψηφίων στην ψηφοφορία. Δηλαδή, στο πιο πάνω σχήμα, φαίνεται ότι στη συγκεκριμένη ψηφοφορία η σειρά κατάταξης των υποψηφίων ήταν: πρώτος ήταν ο Β, δεύτερος ο Α, τρίτος ο C και τέταρτος ο D. Η συνάρτηση αυτή, πριν την εισαγωγή κάθε ονόματος υποψηφίου θα πρέπει να εφαρμόζει αμυντικό προγραμματισμό, ελέγχοντας ότι το όνομα του υποψηφίου που δίνεται, υπάρχει στον πίνακα candidates\_table[] και ότι δεν έχει ξαναδοθεί νωρίτερα στην ίδια ψηφοφορία (άρα δεν υπάρχει ήδη στη λίστα). Για την ψηφοφορία, η συνάρτηση θα πρέπει: α) να υπολογίζει τους βαθμούς του κάθε υποψηφίου, όπως αυτοί προκύπτουν από τη σειρά κατάταξης των υποψηφίων (που φαίνεται στην λίστα) και από το πλήθος των ψηφοφόρων στην ψηφοφορία. β) να ενημερώνει κατάλληλα για κάθε υποψήφιο το πεδίο points του αντίστοιχου κελιού του πίνακα candidates\_table[] στο οποίο αποθηκεύονται αθροιστικά οι συνολικοί βαθμοί κάθε υποψηφίου.

**Γ.** Υλοποιήστε κυρίως πρόγραμμα που θα διαβάζει τα αποτελέσματα Μ ψηφοφοριών για Ν υποψήφιους, όπου Μ, N σταθερές, με επαναλαμβανόμενη κλήση της συνάρτησης Entervotes και θα υπολογίζει το άθροισμα των βαθμών των υποψηφίων για το σύνολο των ψηφοφοριών. Στη συνέχεια θα εμφανίζει τον υποψήφιο που έχει συγκεντρώσει το μεγαλύτερο σύνολο βαθμών και τους βαθμούς του. *Αν δύο ή περισσότεροι υποψήφιοι έχουν το ίδιο μεγαλύτερο σύνολο βαθμών τότε να εμφανίζονται όλοι.*

**Υπόδειξη:** Στο σχέδιο προγράμ­ματος με τίτλο ypoergasia\_4\_code\_template.c η συνάρτηση *main* η οποία περιγράφεται στο υποερώτημα Γ, δίνεται έτοιμη και καλείστε να συμπληρώσετε τον κώδικα των συναρτήσεων που αναφέρθηκαν, *ενώ μπορείτε να ορίσετε και δικές σας βοηθητικές συναρτήσεις.* Για την σύγκριση αλφαριθμητικών προτείνεται η συνάρτηση int strcmp(cs, ct) της βιβλιοθήκης <string.h>.

Ακολουθεί ενδεικτικό παράδειγμα εκτέλεσης για τα δεδομένα του παραδείγματος: Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

**Γενικές Υποδείξεις:**

1. Για τις απαντήσεις της εργασίας μπορείτε να ανατρέξετε στην συμπληρωματική βιβλιογραφία που δίνεται και στα βοηθητικά κείμενα που υπάρχουν στον δικτυακό τόπο / portal της θεματικής ενότητας. Συνιστάται να προσθέσετε στο τέλος της εργασίας σας κατάλογο βιβλιογραφίας.
2. **Τα αρχεία .c θα πρέπει να μπορούν να εκτελεστούν με τον Dev-C++ compiler.**
3. Τρόπος παράδοσης εργασίας:

Οι απαντήσεις πρέπει να είναι γραμμένες με χρήση επεξεργαστή κειμένου (π.χ. Word) σε σελίδες διαστάσεων Α4. Το αρχείο να περιέχει ως πρώτη σελίδα το κείμενο του Εντύπου Υποβολής - Αξιολόγησης και ως δεύτερη σελίδα τον τίτλο «Σχόλια προς τον φοιτητή» (θα συμπληρωθεί από τον καθηγητή σας). Οι απαντήσεις στις υπο-εργασίες θα αρχίζουν από την τρίτη σελίδα, χωρίς να επαναλαμβάνονται οι εκφωνήσεις. Κάθε υπο-εργασία θα αρχίζει από νέα σελίδα. Για την απάντησή σας θα πρέπει να χρησιμοποιήσετε υποχρεωτικά το **Πρότυπο Υποβολής Γραπτής Εργασίας**. Ενδεικτικά, οι άριστες απαντήσεις μπορούν να επιτευχθούν σε περίπου 12-13 σελίδες**. Στο κείμενο της εργασίας θα πρέπει να βάλετε screenshots από την εκτέλεσή του.** Μαζί με το αρχείο κειμένου θα πρέπει να παραδώσετε και τα .c αρχεία προγραμμάτων.

1. **Η καλή παρουσίαση της εργασίας λαμβάνεται υπόψη στην αξιολόγηση της εργασίας.**
2. Αμυντικός προγραμματισμός και σχολιασμός κώδικα:

Ο κώδικας πρέπει να είναι επαρκώς σχολιασμένος και να χρησιμοποιεί στοιχεία αμυντι­κού προγραμματισμού, όπου αυτό ζητείται. Για την εφαρμογή του αμυντικού προγραμματισμού αρκεί να γίνεται έλεγχος ως προς το εάν μία τιμή που εισάγεται από το χρήστη του προγράμματος βρίσκεται εντός αποδεκτών ορίων (π.χ. να είναι θετική, διαφορετική από το 0, μεγαλύτερη από 20 κ.λπ.). Δεν απαιτείται έλεγχος ως προς το εάν η τιμή ανήκει στο σωστό τύπο δεδομένων (π.χ. ακέραιος, πραγματικός αριθμός, χαρακτήρας κ.λπ.) σύμφωνα με τον τύπο της μεταβλητής στην οποία θα αποθηκευτεί η τιμή αυτή.

1. Τρόπος παράδοσης κώδικα:

Στα ερωτήματα όπου ζητείται υλοποίηση κώδικα σε γλώσσα C, **και στο έγγραφο της απάντησής σας και στο αρχείο του κώδικα** θα πρέπει να δίνεται ολόκληρο το πρόγραμμα, επισημαίνοντας με σχόλια πού απαντάτε κάθε ερώτημα. Για να θεωρούνται οι απαντήσεις ολοκληρωμένες θα πρέπει:

* Ο κώδικας (όπου ζητείται) να είναι επαρκώς σχολιασμένος και ενσωματωμένος μέσα στο .doc αρχείο του Word με τις απαντήσεις σας καθώς *και* σε ξεχωριστό .c αρχείο κειμένου, όχι Word.
* Το όνομα κάθε .c αρχείου να περιλαμβάνει το επώνυμό σας με λατινικούς χαρακτήρες, το χαρακτήρα της υπογράμμισης και τον αριθμό του συγκεκριμένου υποερωτήματος (π.χ. αν το επώνυμό σας είναι Γεωργίου, τότε ο κώδικας για την υποεργασία 1β θα έχει το όνομα Georgiou\_1b.c). Κάθε αρχείο C που θα παραδοθεί θα πρέπει τουλάχιστον να περνάει τη φάση της μεταγλώττισης χωρίς λάθη.
* **Στο έγγραφο Word με τις απαντήσεις σας, να περιλαμβάνονται ενδεικτικά screenshots με μαύρα γράμματα σε λευκό φόντο, όπως στις παρούσες εκφωνήσεις**.

Όλα τα .c αρχεία με τον πηγαίο κώδικα και το .doc αρχείο κειμένου να υποβληθούν στη διεύθυνση [**http://study.eap.gr**](http://study.eap.gr)

1. Σύμφωνα με τον Κανονισμό Σπουδών, η καταληκτική ημερομηνία για την παραλαβή της Γ.Ε. από το μέλος ΣΕΠ είναι η επόμενη Τετάρτη από το τέλος της εβδομάδας παράδοσης Γ.Ε. [↑](#footnote-ref-1)
2. https://en.wikipedia.org/wiki/Borda\_count [↑](#footnote-ref-2)