Εργασία 1 – Κρεμάλα

Αναπτύξτε ένα πρόγραμμα που υλοποιεί ένα παιχνίδι κρεμάλας ανάμεσα στον υπολογιστή και έναν παίκτη όπου ο υπολογιστής "κλέβει" με βάση το γράμμα που δίνει ο παίκτης σε κάθε γύρο, με στόχο να καθυστερήσει την ανεύρεση της λέξης. Η επιθυμητή λειτουργικότητα περιγράφεται παρακάτω.

Περιγραφή παιχνιδιού

Στο κλασικό παιχνίδι κρεμάλας, ο υπολογιστής επιλέγει από ένα λεξικό μια λέξη και σε κάθε γύρο την εμφανίζει με παύλες στις θέσεις των γραμμάτων που δεν έχει μαντέψει ακόμη ο παίκτης. Ο παίκτης προσπαθεί να μαντέψει όλα τα γράμματα πριν τελειώσουν οι γύροι του παιχνιδιού.

Το ζητούμενο είναι να φτιαχτεί μια παραλλαγή όπου ο υπολογιστής κλέβει ως εξής:

Δεν επιλέγει μία συγκεκριμένη λέξη στην αρχή, αλλά διατηρεί όλες τις πιθανές λέξεις που ταιριάζουν με τις επιλογές του παίκτη μέχρι στιγμής. Σε κάθε γύρο,

1. Ο παίκτης μαντεύει ένα γράμμα, π.χ. `E`
2. Ο υπολογιστής χωρίζει το λεξικό σε ομάδες ("κλάσεις ισοδυναμίας") με κριτήριο το αν και πού εμφανίζεται το γράμμα σε κάθε λέξη.
3. Κάθε κλάση ισοδυναμίας περιγράφεται με έναν εκπρόσωπο που αντικατοπτρίζει την κοινή ιδιότητα των λέξεων της κλάσης ως προς την θέση ή τις θέσεις του γράμματος που έδωσε ο παίκτης, π.χ. `_E__` για λέξεις 4 γραμμάτων όπου το `E` βρίσκεται μόνο στη 2η θέση.
4. Ο υπολογιστής κρατά την πολυπληθέστερη κλάση κι εμφανίζει τον αντίστοιχο εκπρόσωπο.

Παράδειγμα:

Λεξικό: `ARTS`, `DEAR`, `DEER`, `FEEL`, `FLEW`, `PEAR`, `PEND`, `PERK`, `POOR`, `REAR`

1ος γύρος: Ο παίκτης μαντεύει `E`. Δημιουργούνται οι παρακάτω 4 κλάσεις:

Ιδιότητα	Λέξεις	Εκπρόσωπος	Μέγεθος
χωρίς E	`ARTS, POOR`	`"----"`	*2*
E μόνο στη 2η θέση	`DEAR, PEAR, PEND, PERK, REAR`	`"_E__"`	*5*
E στη 2η και 3η θέση	`DEER, FEEL`	`"_EE_"`	*2*
E μόνο στην 3η θέση	`FLEW`	`"__E_"`	*1*

Επιλογή: `"_E__"` (πολυπληθέστερη). Διατηρείται η δεύτερη κλάση και παραμένουν 5 λέξεις στο παιχνίδι.

2ος γύρος: Ο παίκτης μαντεύει `R`. Δημιουργούνται οι παρακάτω 4 κλάσεις:

Ιδιότητα	Λέξεις	Εκπρόσωπος	Μέγεθος
χωρίς R	`PEND`	`"-E--"`	*1*
R μόνο στη 4η θέση	`DEAR, PEAR`	`"_E_R"`	*2*
R μόνο στην 3η θέση	`PERK`	`"_ER_"`	*1*
R στην 1η και 4η θέση	`REAR`	`"RE_R"`	*1*

Επιλογή: `"_E_R"`. Διατηρείται η δεύτερη κλάση και παραμένουν 2 λέξεις στο παιχνίδι.

3ος γύρος: Ο παίκτης μαντεύει `F`. Καμία λέξη δεν έχει `F` οπότε παραμένει η ίδια κλάση.

Το παιχνίδι συνεχίζει με τον ίδιο τρόπο. Αν απομείνει μια λέξη και ο παίκτης βρει όλα τα γράμματα της, κερδίζει. Διαφορετικά, αν τελειώσουν οι γύροι χωρίς να έχει βρει τη λέξη ο παίκτης, τότε χάνει και ο υπολογιστής εμφανίζει την πρώτη (λεξικογραφικά μικρότερη) λέξη της τρέχουσας κλάσης ως την "επιλεγμένη".

Στο τέλος, δίνεται η δυνατότητα να ξεκινήσει νέο παιχνίδι από την αρχή.

Προθεσμίες / Υποβολή

Η εργασία θα υλοποιηθεί σε δύο στάδια τα οποία θα υποβάλετε ξεχωριστά στο autolab.

Βεβαιωθείτε ότι η εργασία σας πληροί τις απαιτήσεις υλοποίησης.

Σε κάθε στάδιο υποβάλετε ένα αρχείο με όνομα `hw1.c`

1 στάδιο:

- *Προθεσμία: Παρασκευή 27/2/2026, 21:00*
- *Υποβολές*: Έως 10.

Τελική εργασία:

- *Προθεσμία: Κυριακή 8/3/2026, 21:00*
- *Υποβολές*: Έως 10 χωρίς ποινή, -5% για κάθε υποβολή πέρα των 10.
- *Ελάχιστος βαθμός επιτυχίας*: 70%

Δεν υπάρχει δυνατότητα καθυστερημένης υποβολής.

Αρχεία

`libhw1.a`: Στατική βιβλιοθήκη με τις υλοποιήσεις των έτοιμων συναρτήσεων `getWord`, `printErrorMsg`

`hw1.h`: Header file στο οποίο έχουν οριστεί:

- Ένας τύπος enum που αναπαριστά κωδικό λάθους.
- Η επικεφαλίδα της συνάρτησης `getWord`, η οποία παίρνει ως παράμετρο το όνομα ενός αρχείου και κάθε φορά που καλείται επιστρέφει δείκτη προς την επόμενη λέξη που βρίσκεται στο αρχείο (ως δυναμικά δεσμευμένη συμβολοσειρά) ή `NULL` αν το αρχείο δεν περιέχει άλλες λέξεις.
- Η επικεφαλίδα της συνάρτησης `printErrorMsg` η οποία παίρνει ως παράμετρο έναν κωδικό λάθους κι εκτυπώνει αντίστοιχο μήνυμα.

`hw1.c`: Αρχείο C στο οποίο θα υλοποιήσετε την εργασία σας. Περιλαμβάνει την εντολή `#include"hw1.h"`

Ορισμοί

Ορίστε ένα `struct` που αναπαριστά μια κλάση ισοδυναμίας και περιέχει πεδία για τις εξής ποσότητες: (α) τον εκπρόσωπο (δυναμικά δεσμευμένη συμβολοσειρά), (β) το πλήθος λέξεων της κλάσης, (γ) τις λέξεις που ανήκουν στην κλάση (δυναμικά δεσμευμένος πίνακας από δείκτες σε ήδη υπάρχουσες συμβολοσειρές).

Κατά τη διάρκεια του παιχνιδιού θα χρησιμοποιήσετε έναν δυναμικά δεσμευμένο πίνακα από δείκτες προς κλάσεις ισοδυναμίας.

Υλοποίηση, στάδιο 1

Το πρόγραμμα δέχεται τρία ορίσματα: (α) το μήκος της λέξης για την κρεμάλα, (β) το πλήθος γύρων, (γ) το όνομα ενός αρχείου-λεξικού.

Γράψτε μια συνάρτηση ελέγχου των ορισμάτων, που επιστρέφει κωδικό λάθους και παίρνει ως παραμέτρους: (α) το πλήθος ορισμάτων (argc), (β) τον πίνακα ορισμάτων (argv), (γ) δείκτη σε ακέραιο και (δ) δείκτη σε ακέραιο. Η συνάρτηση λειτουργεί ως εξής:

- Αν έχει δοθεί λάθος πλήθος ορισμάτων, επιστρέφει τον κωδικό λάθους `INVALID_ARGS`.
- Αν έχει δοθεί μη-θετικό μήκος λέξης, επιστρέφει τον κωδικό λάθους `INVALID_LEN`.
- Αν έχει δοθεί μη-θετικό πλήθος γύρων, επιστρέφει τον κωδικό λάθους `INVALID_TURNS`.

- Οι παραπάνω έλεγχοι γίνονται με τη σειρά που περιγράφονται. Εάν δεν διαπιστωθεί κάποιο από τα παραπάνω σφάλματα, η συνάρτηση αποθηκεύει στις διευθύνσεις μνήμης που δείχνουν οι 3η και 4η παράμετροι το μήκος της λέξης και το πλήθος γύρων αντίστοιχα, κι επιστρέφει `OK`.

Καλέστε την παραπάνω συνάρτηση στη main. Σε περίπτωση που δεν επιστραφεί `OK`, καλέστε τη συνάρτηση `printErrorMsg` με τον κατάλληλο κωδικό λάθους και τερματίστε το πρόγραμμα.

Γράψτε μια συνάρτηση δημιουργίας του λεξικού που παίρνει ως παραμέτρους (α) το όνομα του αρχείου, (β) το επιθυμητό μήκος λέξης και (γ) τη διεύθυνση ενός ακεραίου. Αυτή η συνάρτηση χρησιμοποιεί τη συνάρτηση `getWord` σε επανάληψη για να διαβάσει τις λέξεις του αρχείου και να τις αποθηκεύσει σε έναν δυναμικά δεσμευμένο πίνακα από δείκτες προς δυναμικά δεσμευμένες συμβολοσειρές. Αν δε βρεθεί καμία λέξη με το επιθυμητό μήκος, η συνάρτηση αποδεσμεύει ό,τι μνήμη τυχόν δέσμευσε κι επιστρέφει `NULL`. Διαφορετικά, αποθηκεύει το πλήθος λέξεων του πίνακα στη θέση μνήμης που δείχνει η 3η παράμετρος κι επιστρέφει τη διεύθυνση στην οποία ξεκινάει ο πίνακας.

Καλέστε την παραπάνω συνάρτηση στη main. Σε περίπτωση αποτυχίας δημιουργίας του λεξικού, εκτυπώστε στη main το μήνυμα "`No words of length X.\n`" όπου `X` το επιθυμητό μήκος και τερματίστε το πρόγραμμα.

Γράψτε μια συνάρτηση δημιουργίας της αρχικής κλάσης ισοδυναμίας, στην οποία ο εκπρόσωπος είναι μια συμβολοσειρά του επιθυμητού μήκους που αποτελείται από κάτω παύλες και ο πίνακας δεικτών περιέχει δείκτες προς όλες τις λέξεις του λεξικού.

Καλέστε την παραπάνω συνάρτηση στη main.

Γράψτε μια συνάρτηση εκτύπωσης των κλάσεων ισοδυναμίας. Η συνάρτηση εκτυπώνει:

- Το μήνυμα "`\n== CLASSES ==\n`"
- Το πλήθος κλάσεων ισοδυναμίας και δύο χαρακτήρες '`\n`'
- Τις κλάσεις ισοδυναμίας, λεξικογραφικά σε αύξουσα σειρά ως προς τους εκπροσώπους. Τα στοιχεία κάθε μίας κλάσης εκτυπώνονται ως εξής:
    - Εκπρόσωπος κλάσης, ένα κενό, το πλήθος λέξεων στην κλάση με πλάτος 3, και ακολούθως οι λέξεις της κλάσης με κενό πριν από κάθε μία, ταξινομημένες λεξικογραφικά σε αύξουσα σειρά, με κεφαλαία γράμματα. Στο τέλος εκτυπώνεται ένας χαρακτήρας αλλαγής γραμμής.

Στη main, εάν είναι ορισμένο το macro DEBUG (δείτε παρακάτω) καλέστε την παραπάνω συνάρτηση.

Υλοποίηση, στάδιο 2

Συνεχίστε την υλοποίηση ως εξής:

Σε επανάληψη μέχρι ο παίκτης να επιλέξει τερματισμό υλοποιήστε το παιχνίδι:

Το παιχνίδι σε κάθε γύρο:

- Εκτυπώνει το μήνυμα "`\n== ROUND X/Y ==\n\n`" όπου `X` ο αύξων αριθμός του γύρου ξεκινώντας από 1, και `Y` το συνολικό πλήθος γύρων.
- Εκτυπώνει το μήνυμα "`Used letters: `" και μετά ένα-ένα τα γράμματα που έχουν ήδη χρησιμοποιηθεί από τον παίκτη, ταξινομημένα σε αύξουσα σειρά, με ένα κενό πριν από κάθε γράμμα και ένα χαρακτήρα '`\n`' μετά το τελευταίο.
- Εκτυπώνει το μήνυμα "`Unused letters: `" και μετά τα υπόλοιπα γράμματα, με τον ίδιο τρόπο.
- Εκτυπώνει το μήνυμα "`\nWord: `" και τη λέξη της κρεμάλας, με χαρακτήρα underscore ('_') στις θέσεις που δεν έχει βρεθεί γράμμα από τον παίκτη, ένα κενό πριν από κάθε γράμμα και '`\n`' στο τέλος.

- Εκτυπώνει το μήνυμα "`\nGuess a letter: `" και διαβάζει ένα χαρακτήρα. Αν αυτός δεν είναι γράμμα, τότε εκτυπώνει το μήνυμα "`X is not a letter.\n`", όπου `X` ο χαρακτήρας που διαβάστηκε, και το βήμα επαναλαμβάνεται. Αν είναι γράμμα αλλά έχει ήδη επιλεγεί σε προηγούμενο γύρο, τότε εκτυπώνει το μήνυμα "`You have already guessed X.\n`", όπου `X` το γράμμα που διαβάστηκε, και το βήμα επαναλαμβάνεται.
- Κατασκευάζει νέες κλάσεις ισοδυναμίας όπως αυτές διαμορφώνονται από την επιλογή γράμματος. Όλες οι κλάσεις ισοδυναμίας πρέπει να είναι αποθηκευμένες σε δυναμικά δεσμευμένο πίνακα από δείκτες προς αυτές ο οποίος ανανεώνεται κατάλληλα από τον ένα γύρο στον επόμενο.
- Εάν είναι ορισμένο το macro `DEBUG`, καλεί τη συνάρτηση εκτύπωσης των κλάσεων ισοδυναμίας.
- Βρίσκει την πολυπληθέστερη κλάση. Σε περίπτωση ισοτιμίας, επιλέγει αυτή που ο εκπρόσωπος της έχει τις περισσότερες παύλες. Σε περίπτωση ισοτιμίας και σε αυτό το κριτήριο, επιλέγει την κλάση με τον λεξικογραφικά "μεγαλύτερο" εκπρόσωπο (ανάμεσα σε αυτές με τις περισσότερες παύλες).
- Εάν είναι ορισμένο το macro `DEBUG`, εκτυπώνει το μήνυμα "`\n== CHOSEN ==\n`" και ακολούθως τα στοιχεία της κλάσης που επιλέχθηκε.
- Ελέγχει αν το παιχνίδι τελείωσε. Αν ο παίκτης κέρδισε, το πρόγραμμα εκτυπώνει το μήνυμα "`\nYou won! The word is X.\n`" όπου `X` η λέξη. Αν έχασε, το πρόγραμμα εκτυπώνει το μήνυμα "`\nYou lost! The word is X.\n`" όπου `X` η λεξικογραφικά μικρότερη λέξη της τρέχουσας κλάσης ισοδυναμίας.

Όταν ένα παιχνίδι τελειώσει, το πρόγραμμα εκτυπώνει το μήνυμα "`\n\nYOU: X - ME: Y\n\n`" όπου `X` ο αριθμός φορών που έχει κερδίσει ο παίκτης και `Y` ο αριθμός φορών που έχει χάσει. Μετά, εκτυπώνει το μήνυμα "`Play again? (y/n)\n`" και διαβάζει ένα χαρακτήρα. Αν αυτός δεν είναι '`y`'/'`Y`'/'`n`'/'`N`', η εκτύπωση του μηνύματος και η ανάγνωση του χαρακτήρα επαναλαμβάνονται. Αν δοθεί '`y`' ή '`Y`' το παιχνίδι ξεκινά από την αρχή (με το ίδιο λεξικό και μήκος λέξης), διαφορετικά τερματίζει. Σε κάθε περίπτωση ελευθερώνει όλη τη δυναμικά δεσμευμένη μνήμη (εκτός του αρχικού λεξικού αν πρόκειται να ξαναξεκινήσει).

Απαιτήσεις υλοποίησης

- Απαγορεύεται η χρήση global ή static μεταβλητών καθώς και η χρήση goto, gets ή fflush.
- Απαγορεύεται η χρήση πινάκων μεταβλητού μεγέθους (variable length arrays).
- Όλες οι κλάσεις ισοδυναμίας που κατασκευάζονται κατά τη διάρκεια του παιχνιδιού πρέπει να περιέχουν δείκτες προς τις ήδη υπάρχουσες συμβολοσειρές του αρχικού πίνακα-λεξικού και όχι αντίγραφα αυτών.
- Όταν το πρόγραμμα δέχεται είσοδο δεν πρέπει να κάνει διάκριση σε κεφαλαία/μικρά. Όταν εκτυπώνει γράμματα, εκπροσώπους ή λέξεις λεξικού, χρησιμοποιεί πάντα κεφαλαία.
- Όλα τα μηνύματα εξόδου που περιέχουν '`:`' έχουν ένα χαρακτήρα space μετά το '`:`'.
- Οι δυναμικά δεσμευμένοι πίνακες πρέπει να έχουν ακριβώς όσο μέγεθος χρειάζεται.
- Σε περίπτωση αποτυχίας δέσμευσης μνήμης εκτυπώνετε το μήνυμα "`Memory error.\n`" και το πρόγραμμα τερματίζει άμεσα με χρήση της εντολής `exit`.
- Φροντίστε να αποδεσμεύετε σωστά όλη τη δυναμικά δεσμευμένη μνήμη πριν το πρόγραμμα τερματίσει ομαλά (αυτό δεν χρειάζεται να γίνει στις περιπτώσεις που το πρόγραμμα τερματίζει μέσω `exit` επειδή εντοπίστηκε κάποιο πρόβλημα).
- Ορίστε συναρτήσεις για συγκεκριμένες λειτουργίες (π.χ. παίξιμο ενός γύρου) καθώς και για λειτουργίες που επαναλαμβάνονται σε διάφορα σημεία (π.χ. εκτύπωση στοιχείων κλάσης)
- Το πρόγραμμα πρέπει να ακολουθεί τους κανόνες μορφοποίησης, σχολιασμού, κτλ. που έχουν αναρτηθεί στο eclass, καθ' όλη τη διάρκεια της ανάπτυξης του.

Μεταγλώττιση & DEBUG

Για να χρησιμοποιήσετε τις συναρτήσεις που ορίζονται στη βιβλιοθήκη `libhw1.a` πρέπει να διασυνδέσετε τον κώδικα σας με αυτή. Αυτό γίνεται προσθέτοντας στην εντολή μεταγλώττισης τις επιλογές: `-lhw1 -L.`

Επιπλέον, υπάρχει η δυνατότητα να ορίσετε ένα όνομα (στην προκειμένη περίπτωση το `DEBUG`) ως macro στο στάδιο της μεταγλώττισης προσθέτοντας την επιλογή `-D DEBUG`. Αυτό είναι ισοδύναμο με το να είχατε την εντολή `#define DEBUG 1` στο πρόγραμμα σας. Ο έλεγχος στον κώδικα για το αν έχει οριστεί το macro `DEBUG` γίνεται ως εξής:

```
#ifdef DEBUG
   /* εδώ μπαίνει ο κώδικας που πρέπει να εκτελεστεί αν το DEBUG έχει οριστεί */
#endif
```

Τελικά, η εντολή μεταγλώττισης σε περίπτωση που επιθυμείτε να έχει οριστεί το DEBUG είναι:

```
gcc -Wall -g -D DEBUG hw1.c -o hw1 -lhw1 -L.
```