ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
Τμήμα Πληροφορικής

ΕΠΛ 232 – Προγραμματιστικές Τεχνικές και Εργαλεία
ΑΣΚΗΣΗ 3 – Υλοποίηση Παιχνιδιού Δόλιας «Κρεμάλας» με Δυναμική Δέσμευση Μνήμης

Διδάσκων: Ανδρέας Αριστείδου

Υπεύθυνος Άσκησης: Παύλος Αντωνίου, Πύρρος Μπράτσκας

Ημερομηνία Ανάθεσης: Τρίτη, 20 Οκτωβρίου 2020
Ημερομηνία Παράδοσης: Κυριακή, 8 Νοεμβρίου 2020 στις 23:59 (19 ημέρες)

(ο κώδικας να υποβληθεί σε zip μέσω του Moodle)

I. Στόχος

Στην εργασία αυτή θα ασχοληθούμε με γραμμικές δομές δεδομένων και δυναμική δέσμευση μνήμης. Το θέμα της άσκησης είναι η υλοποίηση ενός προγράμματος δόλιας κρεμάλας το οποίο δεν χρησιμοποιεί τους συμβατικούς και γνωστούς κανόνες του παιχνιδιού. Η άσκηση αποτελείται από πολλαπλά μέρη. Οι λειτουργίες του προγράμματος και το αναμενόμενο αποτέλεσμα περιγράφονται αναλυτικότερα στην συνέχεια. Για την υλοποίηση της εργασίας αυτής θα πρέπει, πέρα από την χρήση συναρτήσεων και τεχνικών δομημένου προγραμματισμού, να χρησιμοποιήσετε τα ακόλουθα στοιχεία:

1. Διάσπαση του προγράμματος σε πολλαπλά αρχεία .c και .h με χρήση generic **makefile.** Ενδεικτικά αναφέρουμε ότι το πρόγραμμα μπορεί να αποτελείται από μερικά αντικείμενα και αρχεία κεφαλίδας (.c και .h) μαζί με ένα βασικό αρχείο εκκίνησης as3.c.
2. **Κάθε αντικείμενο** πρέπει να συμπεριλαμβάνει τον **σχετικό οδηγό χρήσης** (**driver function**, δείτε διάλεξη 12) και θα πρέπει να έχετε σχόλια και οδηγό σχολίων με χρήση του **doxygen**.

II. Περιγραφή

Δεν είναι εύκολο να γράψει κανείς προγράμματα που υλοποιούν παιχνίδια. Όταν εμείς οι άνθρωποι παίζουμε παιχνίδια, βασιζόμαστε στην εμπειρία του παρελθόντος, προσαρμοζόμαστε στις στρατηγικές των αντιπάλων μας, και μαθαίνουμε από τα λάθη μας.
Οι Ηλεκτρονικοί Υπολογιστές (Η/Υ) από την άλλη πλευρά, ακολουθούν τυφλά ένα προκαθορισμένο αλγόριθμο που (ενδεχομένως) τους αναγκάζει να ενεργήσουν έξυπνα. Σε κάποιες περιπτώσεις παιχνιδιών, όπως στο σκάκι, οι υπολογιστές έχουν επικρατήσει των ανθρώπων. Τα προγράμματα που τρέχουν τέτοιοι Η/Υ βασίζονται σε εκατοντάδες χρόνια ανθρώπινης εμπειρίας, στη χρήση προηγμένων αλγόριθμων, καθώς επίσης και στην τεχνητή νοημοσύνη.

Ενώ υπάρχουν πολλές βιώσιμες στρατηγικές για τη δημιουργία ανταγωνιστικών τεχνικών βάση των οποίων λειτουργεί ο υπολογιστής κατά την διάρκεια του παιχνιδιού, υπάρχει μια εναλλακτική προσέγγιση που στηρίζεται στη δολίευση. Το ερώτημα που τίθεται είναι: *γιατί να περάσετε πάρα πολύ χρόνο προσπαθώντας να «διδάξετε» σε ένα υπολογιστή όλες τις στρατηγικές, όταν μπορείτε απλά να γράψετε ένα πρόγραμμα που παίζει με δόλο και κερδίζει επιδέξια κάθε φορά;*
Αυτό ακριβώς πρέπει να κάνετε σε αυτήν την εργασία. Πρέπει να γράψετε ένα πρόγραμμα που θα παραβλέπει κάποιους κανόνες του παιχνιδιού Κρεμάλα, κάνοντας τον υπολογιστή υπερπαίκτη, δηλαδή να κερδίζει τον ανθρώπινο αντίπαλό του συνεχώς αλλά όχι πάντα.
Σε περίπτωση που δεν είστε εξοικειωμένοι με το παιχνίδι Hangman (Κρεμάλα) σε ηλεκτρονικό υπολογιστή (Η/Υ), οι κανόνες του έχουν ως εξής:

1. Ο Η/Υ επιλέγει μια μυστική λέξη από ένα προκαθορισμένο και συμφωνημένο με τον παίκτη λεξικό, και στη συνέχεια, τυπώνει μια σειρά από παύλες ίση με το μήκος μιας λέξης, στον παίκτη. Παράλληλα ενημερώνει το χρήστη για τον αριθμό των προσπαθειών που έχει στη διάθεση του για να μαντέψει τη μυστική λέξη (ίσως αυτό να συμφωνείται πριν την εκκίνηση του παιχνιδιού μεταξύ του Η/Υ και του παίκτη) .

2. Ο παίκτης ξεκινά να μαντεύει ένα-ένα τα γράμματα με στόχο να βρει τη λέξη που επέλεξε ο Η/Υ. Κάθε φορά που μαντεύει σωστά ένα γράμμα που περιέχεται στην μυστική λέξη, ο Η/Υ αποκαλύπτει κάθε εμφάνιση του εν λόγω γράμματος αφαιρώντας την (τις) αντίστοιχη (αντίστοιχες) παύλα (παύλες). Σε αντίθετη περίπτωση, η εικασία θεωρείται λάθος και μειώνεται κατά ένα ο αριθμός των εναπομενουσών προσπαθειών.
3. Το παιχνίδι τελειώνει είτε όταν όλα τα γράμματα της λέξης έχουν αποκαλυφθεί ή όταν η παίκτης έχει τελειώσει όλες τις προσπάθειες του.

Σημαντικό για το παιχνίδι είναι το γεγονός, ότι σε ένα συμβατικό παιχνίδι κρεμάλας ο Η/Υ έχει προεπιλεγμένη μια συγκεκριμένη λέξη και δεν αλλάζει την λέξη αυτή κατά τη διάρκεια της εκτέλεσης του προγράμματος. Από την άλλη στην εργασία αυτή, ο Η/Υ έχει την ευχέρεια να επιλέγει κατά την εκτέλεση οποιαδήποτε από τις λέξεις του λεξικού και η προσπάθεια του χρήστη θα πρέπει να είναι ο περιορισμός των επιλογών του Η/Υ έτσι ώστε να τον νικήσει.

Ας δούμε τώρα ένα παράδειγμα για να αντιληφθούμε την λογική υλοποίησης. Ας υποθέσουμε ότι ο παίκτης προσπαθεί να μαντέψει τη (μη-επιλεγμένη) λέξη του Η/Υ, και αποκαλύπτοντας γράμματα, αναγκάζει τον Η/Υ να φτάσει στο παρακάτω σημείο (με μια τελευταία προσπάθεια για τον παίκτη):

```
D O _ B L E
```

Αν το συμφωνημένο λεξικό είναι το λεξικό του UNIX (`/usr/share/dict/words`) τότε σε αυτό υπάρχουν μόνο δύο λέξεις ταιριάζουν με αυτό το μοτίβο: "DOABLE" και "DOUBLE". Εάν ο Η/Υ παίζει τίμια, τότε ο παίκτης έχει 50% πιθανότητες να κερδίσει αυτό το παιχνίδι, αν πρόκειται να μαντέψει το «A» ή το «U», σαν το γράμμα που λείπει. Ωστόσο, αν ο Η/Υ τον εξαπατά και στην πραγματικότητα δεν έχει διαλέξει κάποια από τις δυο λέξεις, τότε δεν υπάρχει κανένας πιθανός τρόπος ο παίκτης να κερδίσει αυτό το παιχνίδι αφού ο Η/Υ μπορεί να ισχυριστεί ότι είχε διαλέξει την άλλη λέξη, λέγοντας ότι η εικασία είναι λανθασμένη (και εφόσον έμεινε μόνο 1 προσπάθεια στον παίκτη). Έτσι, ο παίκτης θα χάσει το παιχνίδι. Δηλαδή, αν έχει μαντέψει ότι η λέξη είναι "DOABLE", ο Η/Υ μπορεί να προσποιείται ότι είχε διαλέξει από την αρχή την λέξη "DOUBLE", ή αντίστροφα.

Ας περιγράψουμε αυτή την τεχνική (της δολίευσης) με ένα παράδειγμα. Ας υποθέσουμε ότι παίζεται το παιχνίδι δόλιας κρεμάλας, και είναι η σειρά του Η/Υ να επιλέξει μια λέξη, η οποία θεωρούμε ότι είναι μήκους τεσσάρων χαρακτήρων. Αντί να διαλέξει μια συγκεκριμένη μυστική λέξη, ο Η/Υ δημιουργεί μια λίστα με κάθε λέξη ενός λεξικού που έχει τέσσερα γράμματα. Για να μπορέσουμε να εξηγήσουμε καλυτέρα το παράδειγμα, ας υποθέσουμε ότι ένα δοσμένο λεξικό έχει μόνο λίγες λέξεις με 4 γράμματα, τα οποία παρουσιάζονται παρακάτω:

```
ALLY  BETA  COOL  DEAL  ELSE  FLEW  GOOD  HOPE  IBEX
```

Τώρα, ας υποθέσουμε ότι ο παίκτης μαντεύει το γράμμα **«E»**. Θα πρέπει τώρα ο Η/Υ να πει στον παίκτη ποια γράμματα της λέξης που έχει "διαλέξει" είναι **«E».** **Φυσικά, δεν έχει διαλέξει μια λέξη,** και έτσι έχει πολλαπλές επιλογές για το πού θα αποκαλύψει τα **«E».** Παρακάτω φαίνεται η παραπάνω λίστα λέξεων, με το **«E»** υπογραμμισμένο στην κάθε λέξη:

```
ALLY  BETA  COOL  DEAL  ELSE  FLEW  GOOD  HOPE  IBEX
```

Όπως θα παρατηρήσετε, κάθε λέξη στο λεξικό (ή σε μια λίστα λέξεων), εμπίπτει σε μία από τα πέντε σύνολα λέξεων:

- `_ _ _ _` , το οποίο περιλαμβάνει τις λέξεις `ALLY`, `COOL` και `GOOD`
- `_ _ _ E` , το οποίο περιλαμβάνει τη λέξη `HOPE`
- `_ _ E _` , το οποίο περιλαμβάνει τις λέξεις `FLEW` και `IBEX`
- `_ E _ _` , το οποίο περιλαμβάνει τις λέξεις `BETA` και `DEAL`
- `E _ _ E` , το οποίο περιλαμβάνει τη λέξη `ELSE`

Δεδομένου ότι τα γράμματα που αποκαλύπτει ο Η/Υ πρέπει να αντιστοιχούν σε κάποια λέξη στη λίστα λέξεών του, ο Η/Υ μπορεί να επιλέξει να αποκαλύψει οποιοδήποτε από τα παραπάνω πέντε σύνολα. Υπάρχουν πολλοί τρόποι για να επιλέξει ποιο σύνολο να αποκαλύψει - ίσως θέλει να πάει τον αντίπαλό του (τον παίκτη) προς ένα μικρότερο σύνολο με πιο σπάνιες λέξεις, ή προς ένα μεγαλύτερο σύνολο με την ελπίδα της διατήρησης περισσότερων επιλογών του. **Για αυτήν την εργασία, θα υιοθετήσουμε την τελευταία προσέγγιση και θα επιλέξουμε πάντα το μεγαλύτερο από τα υπόλοιπα σύνολα λέξεων**. Για το παράδειγμα μας, αυτό σημαίνει ότι θα πρέπει ο Η/Υ να επιλέξει το σύνολο _ _ _ _ . Αυτό μειώνει λίστα των λέξεών του σε:

```
ALLY  COOL  GOOD
```

και από τη στιγμή που δεν υπάρχει το γράμμα «**Ε»**, θα πει στον αντίπαλό του ότι η εικασία του ήταν λάθος.
Ας δούμε μερικά ακόμη παραδείγματα υλοποίησης αυτής της στρατηγικής. Λαμβάνοντας υπόψη αυτό το σύνολο τριών λέξεων, αν ο παίκτης μαντέψει το γράμμα **«Ο»**, τότε θα σπάσει η λίστα (το σύνολο) λέξεων σε δύο υπο-σύνολα:

- _ Ο Ο _ , το οποίο περιλαμβάνει τις λέξεις `COOL` και `GOOD`
- _ _ _ _ , το οποίο περιλαμβάνει τη λέξη `ALLY`

Το πρώτο από αυτά τα σύνολα είναι μεγαλύτερο από το δεύτερο, και έτσι ο Η/Υ μπορεί να το επιλέξει, αποκαλύπτοντας δύο **«Ο»** στην λέξη και μειώνοντας τη λίστα του σε: `COOL` και `GOOD`, δηλαδή:

- _ Ο Ο _ , το οποίο περιλαμβάνει τις λέξεις `COOL` και `GOOD`

Τέλος, υπάρχουν δύο πιθανά αποτελέσματα αυτού του παιχνιδιού.

- Κατ 'αρχάς, ο παίκτης μπορεί να είναι αρκετά έξυπνος για να χωρίσει τη λίστα των 2 λέξεων προς μία λέξη και τελικά να μαντέψει αυτή τη λέξη. Σε αυτή την περίπτωση, θα πρέπει να λάβει συγχαρητήρια - αυτό είναι ένα εντυπωσιακό επίτευγμα λαμβάνοντας υπόψη την δολοπλοκία του Η/Υ!
- Δεύτερον, και κατά πολύ η συνηθέστερη περίπτωση, ο παίκτης θα χαθεί εντελώς και θα ξεμείνει από εικασίες (λόγω μπλεξίματος) ή από προσπάθειες (λόγω λανθασμένων επιλογών). Όταν συμβαίνει αυτό, ο Η/Υ μπορεί να επιλέξει οποιαδήποτε λέξη θέλει από τη λίστα του και να πει ότι είναι η λέξη που είχε επιλέξει καθ 'όλη τη διάρκεια του παιχνιδιού. Η ομορφιά αυτής της στρατηγικής είναι ότι ο παίκτης δεν θα έχει καμία δυνατότητα να γνωρίζει ότι ο Η/Υ παρακάμπτει εικασίες κάθε φορά - αυτό μοιάζει απλά με την περίπτωση που ο Η/Υ διάλεξε μια ασυνήθιστη λέξη και την κράτησε μέχρι τέλους παιχνιδιού. ☺

III. Ζητούμενα Άσκησης

Ζητούμενο αυτής της άσκησης είναι η κατασκευή ενός προγράμματος στη γλώσσα προγραμματισμού C, το οποίο να υλοποιεί το παιχνίδι που περιγράφεται στις ενότητες I-II αυτής της εκφώνησης. Το πρόγραμμα σας πρέπει να υλοποιεί τον παρακάτω αλγόριθμο:

1. Διαβάζει από τα ορίσματα γραμμής εντολών (command line arguments) το αρχείο που αντιπροσωπεύει το λεξικό. Υποθέτουμε ότι αρχείο περιέχει μόνο λέξεις με τον ίδιο αριθμό γραμμάτων. Στην πρώτη γραμμή του αρχείου υπάρχει ένας ακέραιος που προσδιορίζει το μήκος των λέξεων αυτού του αρχείου.

2. Ζητά από το χρήστη να εισάγει έναν αριθμό προσπαθειών, ο οποίος πρέπει να είναι ένας ακέραιος αριθμός μεγαλύτερος από το μηδέν (να γίνουν σχετικοί ελέγχοι). Μην ανησυχείτε για το αν δοθεί ένας μεγάλος αριθμός προσπαθειών, καθώς επιλέγοντας περισσότερες από 26 προσπάθειες (όσα τα γράμματα του λατινικού αλφαβήτου) δεν κερδίζει κάτι ο αντίπαλός σας!

3. Το πρόγραμμα να εμφανίζει ένα μήνυμα που παρουσιάζει τον αριθμό των λέξεων (από τη λίστα λέξεων) που έχουν τους συγκεκριμένους χαρακτήρες που έχει μαντέψει σωστά ο παίκτης.

4. Παίξτε το παιχνίδι όπως περιγράφεται παρακάτω:

    a. Εκτυπώνει πόσες **προσπάθειες έμειναν στον παίκτη**, και τη λέξη που πρέπει να μαντέψει η οποία να συμπεριλαμβάνει τα γράμματα που ο παίκτης έχει μαντέψει σωστά και τις γραμμές (_). Εμφανίζει και το μήνυμα του βήματος 3.

b. Προτρέπει το χρήστη να εισάγει (μαντέψει) ένα γράμμα, ζητώντας από τον χρήστη να εισάγει ένα γράμμα το οποίο δεν έχει δώσει προηγουμένως. Βεβαιωθείτε ότι η είσοδος του χρήστη είναι ακριβώς ένας χαρακτήρας και ότι είναι ένα γράμμα του αλφαβήτου. Εναλλακτικά να δίνεται μήνυμα λάθους και ο παίχτης να προσπαθεί ξανά.

c. Αφότου ο παίχτης μαντέψει ένα γράμμα:

   i. το πρόγραμμα χωρίζει τις λέξεις που πήρε από το λεξικό σε σύνολα λέξεων.
   ii. Βρίσκει και διαλέγει το πιο μεγάλο σύνολο λέξεων, αφαιρεί όλες τις άλλες λέξεις από την λίστα λέξεων που δεν βρίσκονται σε αυτό το σύνολο, και αναφέρει τη θέση των γραμμάτων (εάν υπάρχουν) προς το χρήστη. Αν το σύνολο που διάλεξε δεν περιέχει καμία λέξη που να έχει τον εισαγόμενο χαρακτήρα, να αφαιρέσει ένα από τις προσπάθειες του παίκτη.

d. Αν ο παίκτης έχει ξεμείνει από προσπάθειες, το πρόγραμμα να επιλέξει μια λέξη από τη λίστα λέξεων και να εμφανιστεί ως η λέξη που ο υπολογιστής "επέλεξε" από την αρχή.

e. Αν ο παίκτης μαντέψει σωστά όλη τη λέξη, να εμφανιστεί ένα μήνυμα (δείτε παραδείγματα εκτέλεσης πιο κάτω).

Το πρόγραμμα που θα γράψετε διατηρεί μια μεγάλη ψευδαίσθηση: *προσποιείται ότι παίζει ένα παιχνίδι κρεμάλας, αλλά, αντίθετα, κάνει κάτι άλλο στο παρασκήνιο. Κατά συνέπεια, θα πρέπει να προσπαθήσετε να κάνετε το πρόγραμμά σας να* ***ανταποκρίνεται όσο το δυνατόν γρηγορότερα****. Αν οι παίκτες πρέπει να περιμένουν τρία ή περισσότερα δευτερόλεπτα μετά την είσοδό ενός χαρακτήρα, είναι σχεδόν βέβαιο ότι ο παίχτης θα υποπτευθεί ότι κάτι πηγαίνει στραβά.*

IV. Θέματα Υλοποίησης

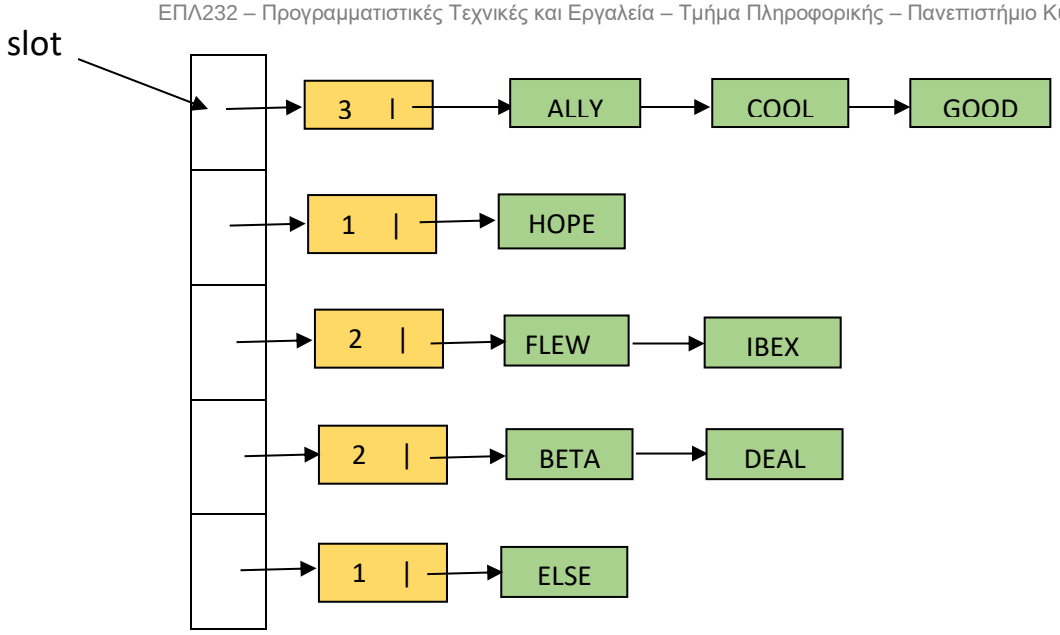
Για αυτήν την εργασία θα πρέπει να χρησιμοποιήσετε γραμμικές δομές δεδομένων όπως πίνακες κατακερματισμού και λίστες. Η δομή μπορεί να είναι η ακόλουθη:

```
typedef struct node {
  char *name;
  struct node *next;
} NODE;


typedef struct list {
  NODE* head;
  int size;
} LIST;


typedef struct {
  LIST **slot;
} HASHTABLE;
```

Κατά τον χωρισμό των λέξεων σε σύνολα, οι λέξεις θα μπουν στην δομή `HASHTABLE`. Για το παράδειγμα που δώσαμε πιο πάνω στην ενότητα II η δομή αυτή αναπαριστάνεται στο παρακάτω σχήμα:



Εικόνα 1. Γραμμική Δομή Δεδομένων για επίλυση του προβλήματος διαχωρισμού σε σύνολα λέξεων (με κίτρινο είναι η δομή LIST και με πράσινο είναι η δομή NODE)

Κάθε φορά που δημιουργείτε σύνολα λέξεων, αυτά πρέπει να αποθηκεύονται σε μια δομή όπως στο παραπάνω σχήμα. Επειδή το λεξικό, άρα και τα σύνολα λέξεων, μπορεί να αποτελείται από πολλές λέξεις, φροντίστε να διαχειριστείτε ορθά την μνήμη.

Η δομή δεδομένων που παρουσιάζεται στην **Εικόνα 1**, είναι μια δομή δεδομένων αποτελούμενη από ένα δυναμικά δεσμευόμενο πίνακα του οποίου τα στοιχεία είναι δείκτες σε λίστες. Για την δημιουργία της δομής αυτής χρησιμοποιείστε και μια συνάρτηση κατακερματισμού, *hash function*, που να μετατρέπει αποτελεσματικά ένα κλειδί (π.χ. λέξεις) στις σχετικές με αυτόν τιμές. **Στην εργασία μας, η συνάρτηση *hash* θα χρησιμοποιείται για να χωριστούν οι λέξεις του λεξικού σε σύνολα λέξεων (π.χ. στο πιο πάνω σχήμα έχουμε σύνολα λέξεων).** Συγκεκριμένα, θα πρέπει να βρείτε ένα τρόπο για την μετατροπή μιας λέξης σε ένα ακέραιο αριθμό που θα αντιπροσωπεύει μια θέση στον πίνακα (τη θέση της λίστας ή του συνόλου στην οποία ανήκει η λέξη). Μετά πρέπει να εισάγετε τη λέξη στην λίστα (σύνολο) που αντιστοιχεί σε αυτήν τη θέση του πίνακα.

Επίσης, οι ακόλουθες συμβουλές και τεχνάσματα μπορούν να σας φανούν χρήσιμα:

- Η **θέση ενός γράμματος** σε μια λέξη είναι τόσο σημαντική όσο είναι και το πλήθος αυτού του γράμματος στην λέξη. Κατά τον υπολογισμό του σε ποιο σύνολο πάει μια λέξη, δεν αρκεί να μετρήσετε πόσες φορές ένα συγκεκριμένο γράμμα εμφανίζεται σε μια λέξη, αλλά πρέπει επίσης, να λάβετε υπόψη και τις θέσεις του γράμματος στην λέξη. Για παράδειγμα, οι λέξεις "BEER" και "HERE" θα βρίσκονται σε δύο διαφορετικά σύνολα, παρόλο που και οι δύο έχουν δύο E. Βεβαιωθείτε ότι η αναπαράσταση των συνόλων λέξεων είναι σωστή ως προς την παραπάνω διάκριση.
- Αν επεξεργάζεστε λέξεις μήκους n, τότε υπάρχουν 2^n πιθανά σύνολα λέξεων για κάθε γράμμα[1]. Ωστόσο, οι περισσότερες από αυτές τις οικογένειες δεν υπάρχουν στην πραγματικότητα στην αγγλική γλώσσα. Για παράδειγμα, δεν υπάρχουν αγγλικές λέξεις που περιέχουν τρία συνεχόμενα U, και δεν υπάρχει λέξη που ταιριάζει με το μοτίβο E _ E E _ E E _ _ E. Αντί να δημιουργήσετε κάθε σύνολο λέξεων κάθε φορά που ο χρήστης εισάγει ένα χαρακτήρα, δείτε αν μπορείτε να δημιουργήσετε σύνολα λέξεων μόνο για τις λέξεις που εμφανίζονται στην πραγματικότητα στη λίστα λέξεων.

[1] Π.χ. Για n=3 (λέξεις με 3 γράμματα) έχουμε τα 8 πιθανά σύνολα _ _ _ (το γράμμα δεν υπάρχει στην λέξη) _ _ X (το γράμμα υπάρχει στη $3^η$ θέση της λέξης), _ X _ , _X X , X_ _ , X _ X , X X _ , X X X (κ.ο.κ)

V. Γενικές Οδηγίες

Το πρόγραμμα σας θα πρέπει να συμβαδίζει με το πρότυπο ISO C, να περιλαμβάνει εύστοχα και περιεκτικά σχόλια, να έχει καλή στοίχιση και το όνομα κάθε μεταβλητής, σταθεράς, ή συνάρτησης να είναι ενδεικτικό του ρόλου της. **Να χρησιμοποιήσετε το λογισμικό τεκμηρίωσης doxygen** έτσι ώστε να μπορούμε να μετατρέψουμε τα σχόλια του προγράμματός σας σε HTML αρχεία και να τα δούμε με ένα browser. Η συστηματική αντιμετώπιση της λύσης ενός προβλήματος περιλαμβάνει στο παρόν στάδιο τη διάσπαση του προβλήματος σε μικρότερα ανεξάρτητα προβλήματα που κατά κανόνα κωδικοποιούμε σε ξεχωριστές συναρτήσεις. Για αυτό τον λόγο σας καλούμε να κάνετε χρήση συναρτήσεων και άλλων τεχνικών δομημένου προγραμματισμού που διδαχτήκατε στο ΕΠΛ131. Επίσης, σας θυμίζουμε ότι κατά την διάρκεια της εκτέλεσης του προγράμματος σας αυτό θα πρέπει να δίνει τα κατάλληλα μηνύματα σε περίπτωση λάθους. Τέλος το πρόγραμμα σας θα πρέπει να μεταγλωττίζεται στις μηχανές του εργαστηρίου. **Παρακαλώ όπως μελετηθούν ξανά οι κανόνες υποβολής εργασιών όπως αυτοί ορίζονται στο συμβόλαιο του μαθήματος.**

Επίσης να ακολουθήσετε τα πιο κάτω βήματα όταν υποβάλετε την άσκηση σας στο Moodle:

1. Δημιουργείστε ένα κατάλογο με το όνομά σας π.χ., PavlosAntoniou/ χωρίς να αφήνετε κενά στο όνομα του καταλόγου.
2. Θυμηθείτε στο configuration file (INPUT =) να συμπεριλάβετε όλα τα αρχεία .h και .c και το README.dox. Ονομάστε το configuration file με το όνομα **as3.conf**
3. Βάλτε μέσα στον κατάλογο αυτό όλα τα αρχεία της εργασίας (κώδικας, as3.conf, README.dox) που πρέπει να υποβάλετε. Μην υποβάλετε αρχείο makefile.
4. Συμπιέστε (zip) τον κατάλογο (και όχι τα αρχεία ξεχωριστά) χρησιμοποιώντας την εντολή *zip –r PavlosAntoniou.zip PavlosAntoniou/**
5. **Βεβαιωθείτε ότι κάνατε σωστά τα προηγούμενα βήματα**
6. Ανεβάστε στο Moodle το συμπιεσμένο αρχείο π.χ., PavlosAntoniou.zip

VI. Κριτήρια αξιολόγησης

Υλοποίηση Δυναμικών Δομών Δεδομένων του Παιχνιδιού	30
Χωρισμός Λέξεων σε Σύνολα Λέξεων	35
Έλεγχοι και Εκτύπωση σωστών μηνυμάτων	15
Διαχωρισμός Προγράμματος σε Πολλαπλά Αρχεία	10
Γενική εικόνα (ευανάγνωστος κώδικας, σχολιασμός doxygen, κλπ.)	10
ΣΥΝΟΛΟ	100

VII. Παράδειγμα εκτέλεσης (τα αρχεία θα τα βρείτε στο as3-supplementary.zip)

Παράδειγμα 1:

```
$./hangMan toyexample.txt
************************************************
***          Welcome to (Evil) Hangman         ***
*** You are going to play against the computer ***
************************************************

How many total guesses? 100

--------------------------------
Progress: ____
Number of Guesses: 100
Guess a letter: E
E not in secret word

--------------------------------
Words possible: 3
Progress: ____
Number of Guesses: 99
Guess a letter: O
You guessed a letter correctly

--------------------------------
Words possible: 2
Progress: _OO_
Number of Guesses: 99
Guess a letter: G
G not in secret word

--------------------------------
Words possible: 1
Progress: _OO_
Number of Guesses: 98
Guess a letter: C
You guessed a letter correctly

--------------------------------
Words possible: 1
Progress: COO_
Number of Guesses: 98
Guess a letter: L
You guessed a letter correctly

************************************************
Congratulations, you win the game
```

```
$ cat toyexample.txt
4
ALLY
BETA
COOL
DEAL
ELSE
FLEW
GOOD
HOPE
IBEX
```

```
The secret word is: COOL
************************************************
```

Παράδειγμα 2:

$./hangMan length4Test.txt

```
************************************************
***          Welcome to (Evil) Hangman         ***
*** You are going to play against the computer ***
************************************************

How many total guesses? 8

--------------------------------
Progress: ____
Number of Guesses: 8
Guess a letter: e
e not in secret word

--------------------------------
Words possible: 13
Progress: ____
Number of Guesses: 7
Guess a letter: a
You guessed a letter correctly

--------------------------------
Words possible: 6
Progress: a___
Number of Guesses: 7
Guess a letter: h
h not in secret word

--------------------------------
Words possible: 5
Progress: a___
Number of Guesses: 6
Guess a letter: t
t not in secret word

--------------------------------
Words possible: 3
Progress: a___
Number of Guesses: 5
Guess a letter: c
You guessed a letter correctly

--------------------------------
Words possible: 3
Progress: ac__
Number of Guesses: 5
Guess a letter: a
Letter a was given before

--------------------------------
Words possible: 3
Progress: ac__
Number of Guesses: 5
Guess a letter: d
```

$ cat length4Test.txt

```
4
acer
ache
achy
acid
acis
acta
acts
actu
acyl
adad
adai
adam
adar
adat
adaw
elsa
else
emil
emim
flew
flex
feat
feck
feed
feel
```

```
d not in secret word

------------------------------
Words possible: 2
Progress: ac__
Number of Guesses: 4
Guess a letter: i
i not in secret word

------------------------------
Words possible: 1
Progress: ac__
Number of Guesses: 3
Guess a letter: y
You guessed a letter correctly

------------------------------
Words possible: 1
Progress: acy_
Number of Guesses: 3
Guess a letter: l
You guessed a letter correctly

**************************************************
Congratulations, you win the game
The secret word is: acyl
**************************************************
```

Παράδειγμα 3:

```
$./hangMan length4Test.txt

**************************************************
***          Welcome to (Evil) Hangman         ***
*** You are going to play against the computer ***
**************************************************

How many total guesses? 7

------------------------------
Progress: ____
Number of Guesses: 7
Guess a letter: e
e not in secret word

------------------------------
Words possible: 13
Progress: ____
Number of Guesses: 6
Guess a letter: a
You guessed a letter correctly

------------------------------
Words possible: 6
Progress: a___
Number of Guesses: 6
Guess a letter: w
w not in secret word
```

```
------------------------------
Words possible: 6
Progress: a___
Number of Guesses: 5
Guess a letter: q
q not in secret word

------------------------------
Words possible: 6
Progress: a___
Number of Guesses: 4
Guess a letter: c
You guessed a letter correctly

------------------------------
Words possible: 6
Progress: ac__
Number of Guesses: 4
Guess a letter: j
j not in secret word

------------------------------
Words possible: 6
Progress: ac__
Number of Guesses: 3
Guess a letter: k
k not in secret word

------------------------------
Words possible: 6
Progress: ac__
Number of Guesses: 2
Guess a letter: l
l not in secret word

------------------------------
Words possible: 5
Progress: ac__
Number of Guesses: 1
Guess a letter: b
b not in secret word

**********************************************
You run out of guesses. You lost
The secret word was: achy
**********************************************
```

Καλή Επιτυχία!