Κ08 Δομές Δεδομένων και Τεχνικές Προγραμματισμού (Τμήμα Φοιτητών/τριών με Άρτιο ΑΜ)

Διδάσκων: Μανόλης Κουμπαράκης

Εαρινό Εξάμηνο 2023-2024

Εργασία 1 (ανακοινώθηκε στις 11 Μαρτίου 2024, προθεσμία παράδοσης: 14 Απριλίου 2024, ώρα 23:59)

1.6 μονάδες του συνολικού βαθμού στο μάθημα. Άριστα=235 μονάδες (υπάρχουν 170 μονάδες bonus, σύνολο 405 μονάδες)

Προσοχή: Πριν διαβάσετε παρακάτω, διαβάστε παρακαλώ προσεκτικά τις οδηγίες υποβολής των ασκήσεων που βρίσκονται στην ιστοσελίδα http://cgi.di.uoa.gr/~k08/homework.html, ειδικά ότι αναφέρεται στο github και τα σχετικά αρχεία.

Απορίες: Αν έχετε απορίες σχετικά με την εργασία, ρωτήστε στο piazza και όχι στέλνοντας e-mail στον διδάσκοντα. Τέτοια e-mails ΔΕΝ θα λαμβάνουν απάντηση.

Κώδικας: Ο κώδικας που παρουσιάσαμε στις διαλέξεις του μαθήματος (Ενότητες 1-5) βρίσκεται στο παρακάτω private repository του classroom του μαθήματος: https://github.com/artioi-k08/2024-ergasia1. Για να συνδεθείτε στο repository αυτό, θα πρέπει να χρησιμοποιήσετε το λινκ https://classroom.github.com/a/_nFaqcFQ. Μπορείτε να χρησιμοποιήσετε αυτόν τον κώδικα στην εργασία αυτή όμως θα πρέπει να το γράψετε ξεκάθαρα στα σχόλια του προγράμματος σας.

Όταν συνδεθείτε θα μπορείτε να δείτε το προσωπικό σας repository https://github.com/artioi-k08/2024-ergasia1-<github-your-username> στο οποίο θα δουλέψετε για την Εργασία 1.

Μετά τη σύνδεση σας, στο προσωπικό σας repository, θα βρείτε τον κώδικα που καλύπτει τις Ενότητες 1-5 του μαθήματος οργανωμένο σε κατάλληλους φακέλους. Θα βρείτε επίσης και ένα φάκελο solutions-ergasia1 με υπο-φακέλους question1, question2, …, question11 στους οποίους θα πρέπει να γράψετε τον κώδικα σας για

τα 9 ερωτήματα της εργασίας που θα βρείτε παρακάτω. Παρακαλώ τηρείστε ευλαβικά αυτήν την οργάνωση αλλιώς θα χάσετε 20% του συνολικού βαθμού κατά την βαθμολόγηση.

Κύριο πρόγραμμα: Σε όλες τις παρακάτω προγραμματιστικές ασκήσεις θα πρέπει να υλοποιήσετε και ένα κύριο πρόγραμμα (συνάρτηση `main`) το οποίο θα διαβάζει τα δεδομένα εισόδου, θα επιδεικνύει τη λειτουργικότητα της συνάρτησης σας για κατάλληλα επιλεγμένες εισόδους, και θα πείθει τον βαθμολογητή ώστε να σας βαθμολογήσει με τον υψηλότερο δυνατό βαθμό.

1. Εξηγήστε πως μπορείτε να υλοποιήστε τον αφαιρετικό τύπο δεδομένων `Queue` χρησιμοποιώντας δύο στοίβες. Η απάντηση σας θα πρέπει να είναι σε ένα pdf αρχείο το οποίο θα βάλετε στον φάκελο solutions-ergasia1/question1 του repository σας.

   (5 μονάδες)

2. Γράψτε μια συνάρτηση η οποία θα ταξινομεί τα στοιχεία μιας στοίβας σε αύξουσα τάξη χρησιμοποιώντας μια άλλη στοίβα. **Ταξινομημένη στοίβα:** στο κάτω μέρος της στοίβας θα βρίσκεται το μικρότερο στοιχείο και στην κορυφή της στοίβας θα βρίσκεται το μεγαλύτερο στοιχείο. Δεν μπορείτε να χρησιμοποιήσετε πίνακες.

   (20 μονάδες)

3. Γράψτε ένα πρόγραμμα το οποίο διαβάζει από την είσοδο μια αριθμητική παράσταση χωρίς αριστερές παρενθέσεις και τυπώνει μια ισοδύναμη παράσταση με τις αριστερές παρενθέσεις κατάλληλα τοποθετημένες. Για παράδειγμα, με δεδομένη την είσοδο

   1 + 2) * 3 – 4) * 5 – 6)))

   το πρόγραμμα θα πρέπει να τυπώνει

   ((1 + 2) * ((3 – 4) * (5 – 6))) .

   (20 μονάδες)

4. Στην άσκηση αυτή θα υλοποιήσετε τον αφαιρετικό τύπο δεδομένων **ουρά διπλού άκρου ή deque** (προφέρεται “deck”) χρησιμοποιώντας ενότητες

(modules) της C και κάνοντας καλή απόκρυψη πληροφορίας. Η ουρά διπλού άκρου είναι ένας συνδυασμός στοίβας και ουράς.

Θα πρέπει να ορίσετε μια κατάλληλη δομή της C για την αναπαράσταση μιας ουράς διπλού άκρου, και να υλοποιήσετε τις παρακάτω πράξεις:

- `Create`: Δημιουργεί μια κενή ουρά διπλού άκρου.
- `SizeOf`: Επιστρέφει ένα ακέραιο που είναι ο αριθμός των στοιχείων της ουράς.
- `IsEmpty`: Επιστρέφει 1 ή 0 ανάλογα αν η ουρά είναι κενή ή όχι.
- `Enqueue`: Εισάγει ένα στοιχείο στο τέλος της ουράς.
- `Push`: Εισάγει ένα στοιχείο στην αρχή της ουράς.
- `Pop`: Αφαιρεί και επιστρέφει το στοιχείο που είναι στην αρχή της ουράς.
- `Dequeue`: Αφαιρεί και επιστρέφει το στοιχείο που είναι στο τέλος της ουράς.
- `Print`: Τυπώνει τα στοιχεία της ουράς από την αρχή προς το τέλος.

Υποθέστε ότι τα δεδομένα της ουράς είναι τύπου `int`. Προσέξτε ότι δεν σας δίνουμε τους τύπους που θα χρειαστείτε για τον αφαιρετικό τύπο δεδομένων, ούτε τους τύπους των ορισμάτων για τις παραπάνω συναρτήσεις, αλλά περιμένουμε από εσας να τους ορίσετε με τον καλύτερο δυνατό τρόπο!

Να δώσετε την υπολογιστική πολυπλοκότητα των παραπάνω πράξεων. Η απάντηση σας για την υπολογιστική πολυπλοκότητα θα πρέπει να είναι σε ένα pdf αρχείο το οποίο θα βάλετε στον φάκελο solutions-ergasia1/question4 του repository σας.

(30+5=35 μονάδες)

5. **(Κυκλική λίστα)** Μια κυκλική λίστα έχει το ίδιο είδος κόμβων με μια απλά συνδεδεμένη λίστα. Δηλαδή, κάθε κόμβος είναι ένα `struct` που περιέχει κάποια πεδία δεδομένων και ένα δείκτη σε ένα `struct` του ίδιου τύπου. Όμως, αντί ο τελευταίος κόμβος να έχει δείκτη `NULL`, δείχνει πίσω στον πρώτο κόμβο. Επομένως, δεν υπάρχει πρώτος ή τελευταίος κόμβος. Αν σαρώσουμε τους κόμβους μιας κυκλικής λίστας από οποιοδήποτε κόμβο, ακολουθώντας

τους δείκτες που δείχνουν στον επόμενο κόμβο, θα περάσουμε κυκλικά από όλους τους κόμβους. Αν και μια κυκλική λίστα δεν έχει αρχή ή τέλος, υπάρχει ένας ειδικός κόμβος, που τον λέμε *cursor* (*δρομέα)*, που μας δίνει πρόσβαση στη λίστα.

Να χρησιμοποιήσετε modules της C και καλή απόκρυψη πληροφορίας για να υλοποιήσετε ένα αφαιρετικό τύπο δεδομένων `CircularList` με τις παρακάτω πράξεις:

- `Create`: Δημιουργεί μια κενή κυκλική λίστα. Η τιμή του δρομέα στην περίπτωση αυτή είναι `NULL`.
- `Add`: Εισάγει ένα νέο κόμβο αμέσως μετά τον δρομέα. Αν η λίστα είναι κενή, τότε ο κόμβος αυτός γίνεται δρομέας και δείχνει στον εαυτό του.
- `Remove`: Διαγράφει και επιστρέφει τον κόμβο που είναι άμεσα επόμενος του δρομέα (όχι τον ίδιο τον δρομέα, εκτός αν είναι ο μοναδικός κόμβος). Αν η λίστα γίνει κενή, η τιμή του δρομέα γίνεται `NULL`.
- `Advance`: Προχωράει τον δρομέα στον επόμενο κόμβο της λίστας.

Υποθέστε ότι τα δεδομένα της κυκλικής λίστας είναι τύπου `int`.

(30 μονάδες)

6. **(Το πρόβλημα του Σήφη)** Θεωρήστε το εξής πρόβλημα. Ν Σφακιανοί βρίσκονται σε κίνδυνο και συμφωνούν την εξής στρατηγική για τη μείωση του πληθυσμού τους. Στέκονται σε ένα κύκλο (με τις θέσεις αριθμημένες από 0 έως Ν-1) και προχωρούν κυκλικά, ξεκινώντας από μια συγκεκριμένη θέση Ρ του κύκλου, εξοντώνοντας κάθε Μ-οστό άτομο μέχρι να μείνει μόνο ένα άτομο. Σύμφωνα με τον Σφακιανό θρύλο, ο Σήφης υπολόγισε που έπρεπε να σταθεί προκειμένου να αποφύγει την εξόντωση.

   Γράψτε ένα πρόγραμμα που θα χρησιμοποιεί τον αφαιρετικό τύπο δεδομένων `CircularList` του προηγούμενου ερωτήματος το οποίο παίρνει τρία ορίσματα γραμμής εντολών Ν, Μ και Ρ και τυπώνει τη σειρά εξόντωσης των ανθρώπων (οπότε θα δείξει στον Σήφη που πρέπει να σταθεί στον κύκλο).

   (20 μονάδες)

7. Να υλοποιήσετε μια αναδρομική συνάρτηση που παίρνει ως όρισμα ένα φυσικό αριθμό N και τυπώνει τους 2^N συνδυασμούς των αριθμών $1, 2, \cdots, N$. Για παράδειγμα, όταν $N = 3$, θα πρέπει να πάρετε την ακόλουθη έξοδο:

$$0, 1, 2, 3, 1\ 2, 1\ 3, 2\ 3, 1\ 2\ 3$$

Το 0 στην παραπάνω ακολουθία παριστάνει το κενό σύνολο.

(20 μονάδες)

8. Γράψτε ένα πρόγραμμα που θα παίρνει σαν είσοδο μια ενθεματική (infix) παράσταση αποτελούμενη από ακεραίους (`int`) και τους τελεστές `+`, `-`, `*`, `/`, `++` και `--`, και θα την αποτιμά χρησιμοποιώντας δύο στοίβες. Η λύση σας θα πρέπει να παραδοθεί σαν ένα module της C και θα πρέπει να γίνεται απόκρυψη πληροφορίας με τον καλύτερο δυνατό τρόπο.

(30 μονάδες)

9. Ο αλγόριθμος *ταξινόμησης* ενός πίνακα ακεραίων `A` με *επιλογή (selection sort)* λειτουργεί ως εξής. Πρώτα βρίσκουμε το μικρότερο στοιχείο του πίνακα και το αντιμεταθέτουμε με το στοιχείο που βρίσκεται στην πρώτη θέση του πίνακα. Έπειτα βρίσκουμε το δεύτερο μικρότερο στοιχείο και το αντιμεταθέτουμε με το στοιχείο που βρίσκεται στην δεύτερη θέση, και συνεχίζουμε με αυτό τον τρόπο μέχρι να ταξινομηθεί ολόκληρος ο πίνακας.

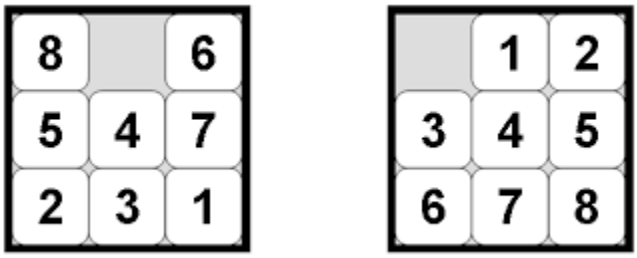
Υλοποιήστε τον παραπάνω αλγόριθμο για να ταξινομήσετε μια απλά συνδεδεμένη λίστα. Μπορείτε να χρησιμοποιήσετε τους ορισμούς τύπων από το πρόγραμμα που παρουσιάστηκε στην πρώτη ενότητα διαλέξεων (Συνδεδεμένες Αναπαραστάσεις Δεδομένων) αφού αλλάξετε των τύπο των στοιχείων της λίστας ώστε να είναι ακέραιοι.

(25 μονάδες)

10. Υλοποιήστε τον αφαιρετικό τύπο δεδομένων Ουρά Προτεραιότητας (Priority Queue) που συζητήσαμε στην Ενότητα 3 των διαλέξεων ώστε να γίνεται καλή απόκρυψη πληροφορίας (information hiding) με χρήση modules της C και handles όπως δείξαμε στα εξής παραδείγματα αφαιρετικών τύπων δεδομένων: Μιγαδικοί Αριθμοί, Στοίβα και Ουρά Αναμονής (χρησιμοποιώντας κώδικα από το βιβλίο του Sedgewick). Γράψτε επίσης και ένα κύριο πρόγραμμα που επιδεικνύει τη λειτουργικότητα της ουράς προτεραιότητας χρησιμοποιώντας την για να ταξινομήσει ένα πίνακα ακεραίων όπως κάναμε στη σχετική διάλεξη.

(30 μονάδες)

11. Στην άσκηση αυτή θα γράψετε ένα πρόγραμμα που θα λύνει το **παζλ των 8 πλακιδίων (8-puzzle)**.



Το παζλ των 8 πλακιδίων ορίζεται ως εξής. Έχουμε στη διάθεση μας ένα πίνακα 3x3 με οκτώ αριθμημένα πλακίδια και μια κενή θέση. Ένα πλακίδιο που γειτονεύει με την κενή θέση μπορεί να μετακινηθεί στη θέση αυτή. Ο σκοπός είναι, ξεκινώντας από ένα δοσμένο πίνακα (π.χ., αυτόν στα αριστερά της παραπάνω εικόνας), να φτάσουμε σε ένα δοσμένο πίνακα (π.χ., αυτόν στα δεξιά της παραπάνω εικόνας).

Το παζλ των 8 πλακιδίων είναι ένα **πρόβλημα αναζήτησης (search problem)**. Τέτοια προβλήματα θα μελετήσουμε με μεγάλη λεπτομέρεια στο μάθημα της Τεχνητής Νοημοσύνης. Ένα πρόβλημα αναζήτησης μπορεί να οριστεί τυπικά με τα εξής στοιχεία:

- **Καταστάσεις.** Στην περίπτωση του παζλ των 8 πλακιδίων, μια κατάσταση καθορίζει τη θέση καθενός από τα 8 πλακίδια και της κενής θέσης.
- **Αρχική κατάσταση.** Οποιαδήποτε κατάσταση μπορεί να οριστεί ως αρχική κατάσταση. Στο παραπάνω παράδειγμα, η αρχική κατάσταση είναι ο αριστερός πίνακας.
- **Ενέργειες.** Το σύνολο των ενεργειών που είναι στη διάθεση του προγράμματος που θα λύσει το πρόβλημα. Για το πρόβλημα των 8 πλακιδίων, είναι βολικό να θεωρήσουμε 4 διαθέσιμες ενέργειες που αντιστοιχούν σε μετακινήσεις της κενής θέσης αριστερά, δεξιά, πάνω και κάτω.[1]

[1] Όταν η κενή θέση μετακινείται στη θέση ενός πλακιδίου, το πλακίδιο καταλαμβάνει τη θέση που ήταν κενή.

- **Συνάρτηση διαδόχων.** Η συνάρτηση αυτή παράγει τις επιτρεπτές καταστάσεις που προκύπτουν από τη εκτέλεση των διαφόρων ενεργειών.
- **Κατάσταση στόχου.** Οποιαδήποτε κατάσταση μπορεί να οριστεί ως κατάσταση στόχου. Στο παραπάνω παράδειγμα, η κατάσταση στόχου είναι ο δεξιός πίνακας.
- **Έλεγχος στόχου.** Ελέγχει αν η κατάσταση στην οποία βρίσκεται το πρόγραμμα που λύνει το πρόβλημα αναζήτησης ταυτίζεται με την κατάσταση στόχου.
- **Κόστος ενέργειας.** Μας δίνει πόσο κοστίζει κάθε ενέργεια. Για το παζλ των 8 πλακιδίων κάθε ενέργεια κοστίζει 1.

Μια **λύση** ενός προβλήματος είναι μια ακολουθία ενεργειών (δηλαδή μετακινήσεων της κενής θέσης στο παζλ των 8 πλακιδίων) που μας επιτρέπει να φτάσουμε από μια δοσμένη αρχική κατάσταση σε μια κατάσταση στόχου. Η **βέλτιστη λύση** του προβλήματος είναι αυτή που έχει το ελάχιστο κόστος (δηλαδή, αυτή που απαιτεί τις λιγότερες μετακινήσεις).

Έχετε να γράψετε μια συνάρτηση η οποία να χρησιμοποιεί επανάληψη (loop) και μια ουρά προτεραιότητας για να λύσει ένα δοσμένο παζλ 8 πλακιδίων και να βρει την βέλτιστη λύση κάνοντας **αναζήτηση**. Η υλοποίηση της ουράς προτεραιότητας που θα χρησιμοποιήσετε θα πρέπει να είναι αυτή του ερωτήματος 10.

Η ουρά προτεραιότητας θα χρησιμοποιείται για να αποθηκεύουμε τις **καταστάσεις** στις οποίες βρίσκεται ο 3x3 πίνακας καθώς το πρόγραμμα σας λύνει το πρόβλημα. Κάθε κατάσταση στην ουρά προτεραιότητας μας περιγράφει με ακρίβεια τη θέση όλων των πλακιδίων και της κενής θέσης και συνοδεύεται από μια **αξιολόγηση** που μας λέει πόσο καλή είναι ώστε, μέσω αυτής, να οδηγηθούμε στην κατάσταση στόχου. Το πρόγραμμα αξιολογεί κάθε κατάσταση k στην οποία μπορεί να βρεθεί ο κόσμος με τη χρήση μιας συνάρτησης αξιολόγησης $f(k) = g(k) + h(k)$. Η συνάρτηση $g(k)$ μας δίνει το

κόστος που έχουμε για να φτάσουμε στην κατάσταση k ξεκινώντας από την αρχική κατάσταση. Η $h(k)$ είναι μια **ευρετική συνάρτηση** η οποία, για το πρόβλημα μας, δίνει τον αριθμό των πλακιδίων που δεν είναι στη σωστή θέση στην κατάσταση k. Αρχικά στην ουρά προτεραιότητας βρίσκεται η αρχική κατάσταση (με την αξιολόγηση της). Ο επαναληπτικός αλγόριθμος λειτουργεί ως εξής. Εξάγουμε το στοιχείο της ουράς με την μεγαλύτερη προτεραιότητα. Ορίζουμε να έχει μεγαλύτερη προτεραιότητα το στοιχείο (κατάσταση) k με το **μικρότερο** $f(k)$. Αν το στοιχείο αυτό αντιστοιχεί σε κατάσταση στόχου, τότε έχουμε τελειώσει και εκτυπώνουμε την ακολουθία κινήσεων που μας οδήγησε από την αρχική κατάσταση στην κατάσταση στόχου. Αν το στοιχείο αυτό δεν αντιστοιχεί σε κατάσταση στόχου, τότε εισάγουμε στην ουρά όλες τις καταστάσεις (μαζί με τις αξιολογήσεις τους) στις οποίες μπορούμε να βρεθούμε από την παρούσα κατάσταση, εκτελώντας μία μόνο ενέργεια (δηλαδή, κίνηση της κενής θέσης). Μετά ο αλγόριθμος συνεχίζει με τον ίδιο τρόπο: εξάγει το στοιχείο της ουράς με τη μεγαλύτερη προτεραιότητα κλπ.

Προτείνουμε να εκτελέσετε αυτό τον αλγόριθμο στο χαρτί για να επιβεβαιώσετε ότι λύνει ένα εύκολο παζλ 8 πλακιδίων και βρίσκει την βέλτιστη λύση, πριν δοκιμάσετε να τον υλοποιήσετε.

Το πρόγραμμα σας θα πρέπει να δουλεύει για οποιοδήποτε πίνακα NxN θέλει ο χρήστης. Θα παρατηρήσετε όμως ότι, όσο καλή κι αν είναι η υλοποίηση σας, ο παραπάνω αλγόριθμος θα καθυστερεί να βρει λύση από ένα αριθμό N και μετά. Ο βασικός λόγος που συμβαίνει αυτό είναι γιατί το πρόβλημα της εύρεσης βέλτιστης λύσης για Μ πλακίδια είναι **NP-complete**[2] δηλαδή δεν γνωρίζουμε γι αυτό αλγόριθμους που τρέχουν σε πολυωνυμικό χρόνο, και πιστεύουμε ότι τέτοιοι αλγόριθμοι δεν υπάρχουν (αν και δεν το έχουμε αποδείξει μαθηματικά – η ερώτηση αυτή είναι μια σπουδαία ανοικτή ερώτηση στην Πληροφορική!). Για να δείτε αυτή τη συμπεριφορά καθαρά, θέλουμε να

[2] Αν ενδιαφέρεστε, δείτε το σχετικό άρθρο https://www.cs.wmich.edu/elise/courses/cs431/icga2008.pdf . Δεν χρειάζεται να το διαβάσετε για την εργασία αυτή! Για NP-complete προβλήματα γενικά, θα μάθετε στο μάθημα Αλγόριθμοι και Πολυπλοκότητα.

υπολογίσετε το **χρόνο εκτέλεσης** του αλγόριθμου σας για N=8, 9, 10, … και να παραδώσετε ένα αρχείο pdf που θα περιέχει ένα πίνακα όπως τον παρακάτω:

N (πάζλ $M=N^2-1$ πλακιδίων)	**Χρόνος Εκτέλεσης σε …**
8	…
8	…
10	…
…	…
…	(η εκτέλεση του προγράμματος διακόπηκε μετά από … λεπτά)

Σε ποιο N μπορέσατε να φτάσετε χωρίς να βαρεθείτε και να τερματίσετε το πρόγραμμα σας; 😊

Το πρόγραμμα σας θα πρέπει να εκτυπώνει την αρχική κατάσταση (αρχικό πίνακα NxN) και μετά την ακολουθία ενεργειών και καταστάσεων που ακολουθούν μέχρι να φτάσουμε στην κατάσταση στόχου.

(100 μονάδες bonus)

Αν βρείτε την άσκηση αυτή ενδιαφέρουσα, μπορείτε να διαβάσετε για **προβλήματα αναζήτησης** και ευρετικές συναρτήσεις από τις διαφάνειες του μαθήματος της Τεχνητής Νοημοσύνης (http://cgi.di.uoa.gr/~ys02/siteAI2020/lectures.html) ή από όποιο άλλο υλικό βρείτε στον παγκόσμιο ιστό, με σκοπό να υλοποιήσετε μια ευρετική συνάρτηση **καλύτερη από την *h*** που δώσαμε παραπάνω. Έτσι θα μπορέσετε να λύσετε σε μικρότερο χρόνο ένα δοσμένο πρόβλημα κύβων. Για να παρουσιάσετε τα αποτελέσματα σας, θα πρέπει να φτιάξετε ένα πίνακα όπως τον παραπάνω που να έχει διαφορετικές στήλες για κάθε μια από τις ευρετικές συναρτήσεις που θα δοκιμάσετε:

N	**Χρόνος Εκτέλεσης σε … με χρήση της ευρετικής *h***	**Χρόνος Εκτέλεσης σε … με χρήση της ευρετικής *h1***
8	…	…

9	…	…
10	…	…
…	…	…
…	(η εκτέλεση του προγράμματος διακόπηκε μετά από … λεπτά)	(η εκτέλεση του προγράμματος διακόπηκε μετά από … λεπτά)

(20 μονάδες bonus)

Αν σας αρέσει ο προγραμματισμός, μπορείτε να φτιάξετε και μια κατάλληλη διεπαφή (interface) όπως αυτήν της ιστοσελίδας https://www.artbylogic.com/puzzles/numSlider/numberShuffle.htm που θα επιτρέπει σε κάποιον να λύσει το παζλ των 8 ή 15 ή … πλακιδίων. Η διεπαφή θα πρέπει να προσφέρει και τη δυνατότητα, να λύσει το παζλ ο αλγόριθμος σας. Για την υλοποίηση της διεπαφής, προτείνεται να χρησιμοποιήσετε τη βιβλιοθήκη raylib (https://www.raylib.com/). Στην ιστοσελίδα της βιβλιοθήκης περιέχονται πολλά παραδείγματα τα οποία συνοδεύονται και από τον αντίστοιχο πηγαίο κώδικα. Μπορείτε να συνδυάσετε στοιχεία από τον κώδικα αυτών των παραδειγμάτων για να σχεδιάσετε τη δική σας διεπαφή. Φυσικά, δεν υπάρχει κάποιος περιορισμός σχετικά με το πως πρέπει να είναι το η διεπαφή σας, αυτό είναι δική σας επιλογή!

(50 μονάδες bonus, παράδοση μέχρι την ημερομηνία παράδοσης της 3ης εργασίας)

Καλή Επιτυχία!