Οικονομικό Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής
Μάθημα: Δομές Δεδομένων
Ακαδημαϊκό έτος: 2019–20
Μέλη: Ελένη Σαξώνη-3160153, Αριστείδης Χρονόπουλος-3160194

Εργασία 1

Μέρος Α:

StringQueue:
Για την υλοποίηση της διεπαφής της ουράς και για να αποφύγουμε το μεγάλο κόστος μνήμης που θα είχε μια υλοποίηση με πίνακες, χρησιμοποιήσαμε λίστα μονής σύνδεσης. Αυτό μας επιτρέπει να κάνουμε τις πράξεις εισαγωγής και εξαγωγής σε χρόνο O(1), ανεξάρτητο από τον αριθμό των αντικειμένων στην ουρά.
Χρησιμοποιώντας τους δείκτες front και rear εκτελούμε σύμφωνα με τη μέθοδο FIFO εισαγωγές στο τέλος της ουράς χρησιμοποιώντας τον rear και εξαγωγές από την αρχή χρησιμοποιώντας τον δείκτη front.
Επομένως η StringQueueImpl περιέχει υλοποιήσεις των παρακάτω μεθόδων:

- `isEmpty(): ελέγχει εάν η ουρά είναι άδεια.`
- `put(String item): εισάγει ένα στοιχείο τύπου String στο τέλος της ουράς.`
- `get(): εξάγει και επιστρέφει ένα στοιχείο από την αρχή της ουράς.`
- `peek(): επιστρέφει το πρώτο στοιχείο της ουράς.`
- `printQueue(): Εμφανίζει τα στοιχεία που περιέχει η ουρά.`
- `size(): Επιστρέφει το μέγεθος της ουράς.`

StringStack:
Για την υλοποίηση της διεπαφής της στοίβας χρησιμοποιήσαμε έναν δείκτη top, για να κρατάμε το τελευταίο στοιχείο που βάλαμε στη στοίβα και εκείνο που θα αφαιρέσουμε πρώτο, βασιζόμενοι στην μέθοδο LIFO.
Επομένως η StringStackImpl περιέχει υλοποιήσεις των παρακάτω μεθόδων:

- `isEmpty(): ελέγχει εάν η στοίβα είναι άδεια.`
- `push(String item):εισάγει ένα στοιχείο τύπου String στην κορυφή της στοίβας.`
- `Pop(): αφαιρεί το πρώτο στοιχείο της στοίβας.`
- `Peek(): επιστρέφει το πάνω πάνω στοιχείο στη στοίβα, χωρίς να το αφαιρεί.`
- `PrintStack(): Εμφανίζει το στοιχεία που περιέχει η στοίβα.`
- `size(): Επιστρέφει το μέγεθος της στοίβας.`

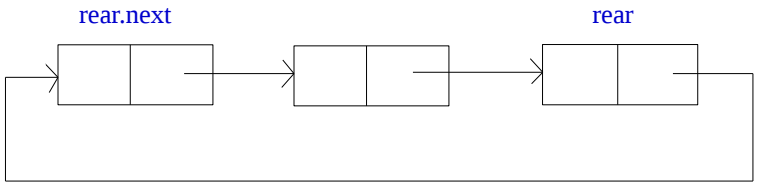
Μέρος Β:

Στο δεύτερο μέρος της εργασίας προκειμένου να βρούμε μια διαδρομή, από ένα αρχείο το οποίο δίνεται αρχικά σαν όρισμα, κάναμε χρήση της στοίβας που υλοποιήσαμε στο πρώτο μέρος. Ορίσαμε μερικές πιθανές κινήσεις που μπορούμε να κάνουμε κατά την εξερεύνηση του μονοπατιού,
0 → πάνω, 1 → αριστερά, 2 → δεξιά, 3 → κάτω. Τυχαία κάθε φορά διαλέγουμε μια κίνηση. Στον boolean πίνακα visited βλέπουμε αν έχουμε ξαναεπισκεφτεί κάποια θέση. Αρχικοποιείται με τιμές true

και ύστερα όταν επισκεπτόμαστε κάποια θέση γίνεται false. Κάθε φορά η νέα θέση στην οποία πηγαίνουμε μπαίνει στην στοίβα. Για να αποφύγουμε πιθανά αδιέξοδα κρατάμε σε μια μεταβλητή hint τις θέσεις που προχωρήσαμε από μια θέση η οποία είχε και άλλες πιθανές διαδρομές, έτσι ώστε σε περίπτωση που δεν μπορούμε να προχωρήσουμε κάνουμε pop από την στοίβα τόσες θέσεις όσες ο μετρητής hint.

Μέρος Γ:

Για την υλοποίηση της ουράς με έναν δείκτη χρησιμοποιήσαμε κυκλική λίστα αντί για λίστα μονής σύνδεσης, το οποίο μας επιτρέπει να κάνουμε χρήση ενός μόνο δείκτη. Επιλέξαμε τον δείκτη rear, το πεδίο next του οποίου δείχνει στην αρχή της ουράς. Επομένως ο δείκτης rear είναι το τελευταίο στοιχείο της ουράς, ενώ ο rear.next είναι το πρώτο στοιχείο. Έτσι μπορούμε να πραγματοποιήσουμε εισαγωγές στο τέλος της ουράς με την χρήση του δείκτη rear και εξαγωγές από την αρχή της ουράς με τον δείκτη rear.next.



Επομένως υλοποιούμε τις παρακάτω μεθόδους ως εξής:

- `put(String item)`: ελέγχουμε εάν η ουρά είναι άδεια, επομένως στο πεδίο next του κόμβου που εισάγουμε αρκεί να δείχνει στον εαυτό του, ενώ αν δεν είναι άδεια αρκεί ο νέος κόμβος να δείχνει στην αρχή(rear.next) και ο τελευταίος να δείχνει πλέον στον καινούριο. Τέλος ο νέος κόμβος γίνεται το τελευταίο στοιχείο της ουράς.
- `get()`: Αν η λίστα δεν είναι άδεια πρέπει να ελέγξουμε αν έχουμε μόνο έναν κόμβο στην ουρά(αν rear.next==rear) τότε επιστρέφουμε τον συγκεκριμένο κόμβο, αλλίως επιστρέφουμε τον πρώτο κόμβο πάντα(rear.next), και φροντίζουμε ο rear.next.next να είναι πλέον ο πρώτος κόμβος της ουράς.