01/12/2020

ΚΩΝΣΤΑΝΤΙΝΟΣ ΜΑΡΚΟ ΑΜ:3190112
ΑΛΕΞΑΝΔΡΟΣ ΚΟΝΟΜΗΣ ΑΜ:3140085

1Η ΕΡΓΑΣΙΑ

ΘΕΜΑ Α

Ξεκινάμε το implementation της StringDoubleEndedQueueImpl<T> στο εσωτερικό της φτιάχνοντας την nested class Node<T> με constructor που παίρνει ως ορίσματα την τιμή του κόμβου(item) καθώς και τους κόμβους που θέλουμε να του συνδέσουμε στην αρχή(previous) και στο τέλος(next).

int size():Ορίζουμε μια μεταβλητή int size που θα μεταβάλλεται στις μεθόδους addFirst() addLast() removeFirst() removeLast().Η πολυπλοκότητα θα είναι O(1) καθώς είτε θα γυρνάει 0(αν η ουρά είναι κενή) είτε το πραγματικό μέγεθος της χωρίς περαιτέρω βήματα(απλή επιστροφή της μεταβλητής size)

T addFirst() & T addLast():Στην addFirst() έχουμε σαν όρισμα μια τιμή την οποία αφού ορίσουμε σε έναν κόμβο την εκχωρούμε στο previous της κεφάλης της ουράς και αυξάνουμε το μέγεθος της κατά 1.Παρομοια στην addLast() με την διαφορά ότι εκχωρούμε τον νέο κόμβο στο next του τελευταίου κόμβου της ουράς(Πολυπλοκότητα 0(1) διότι είτε θα εκχωρήσουμε null σε head και tail λόγω κενής ουράς, είτε εισαγωγή ενός καινούριου κόμβου)

T removeFirst() & removeLast(): Παρόμοια λογική με addFirst και addLast αλλά με "NoSuchElementException" Exception στην περίπτωση που η ουρά μας είναι άδεια. Στη removeFirst το head θα διαγραφθεί και το head.next θα γίνει head και στο removeLast το tail θα αφαιρεθεί και το tail.previous θα γίνει tail(αν δεν είναι κενη η ουρα από την αφαιρεση κομβου) και στις περιπτώσεις το μέγεθος θα μειωθεί.(Πολυπλοκότητα O(1) καθώς είτε exception θα υπάρξει διότι η λίστα θα είναι άδεια είτε απλή αφαίρεση ενός κόμβου).

T getFirst() & getLast():Απλή επιστροφή head.item (getFirst) tail.item (getLast) εφόσον η ουρά δεν είναι κενή.O(1) ανεξαρτήτως απ' το μέγεθος ουράς.

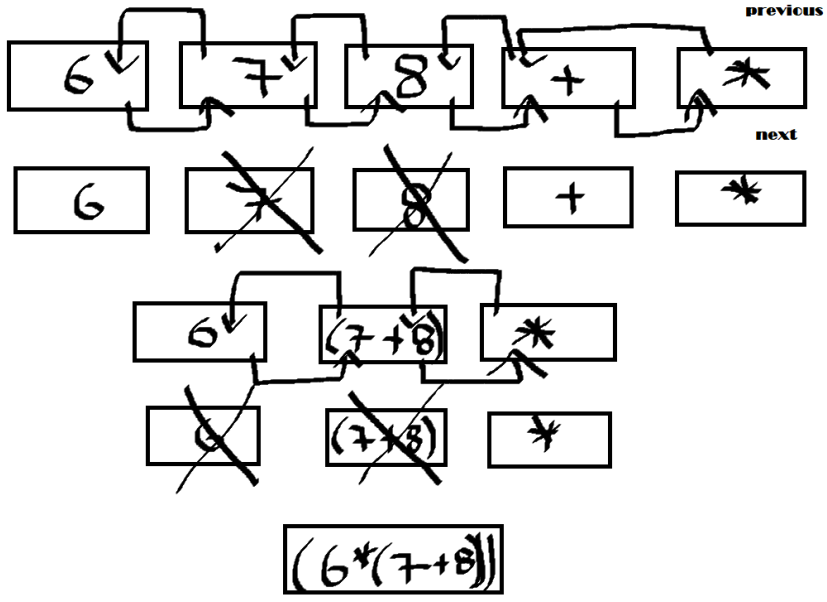
Boolean isEmpty():Επιστροφή true αν το size διάφορο του 0 αλλιώς false.

void printQueue():Αρχικοποίηση του head για την επανάληψη και εμφάνιση των στοιχείων μέσω της κλάσης PrintStream.

ΘΕΜΑ Β

Ξεκινώντας χρησιμοποιούμε την **Scanner** class,διαβάζουμε τον String postfix ,δημιουργούμε αντικείμενο ουράς και με επανάληψη το κάθε char του postfix το μετατρέπουμε σε String και αφού ελέγξουμε ότι όλα τα στοιχεία είναι αριθμοί απ' το 0-9 ή τελεστές (+,-,*,/) και ότι το τελευταίο στοιχείο είναι τελεστής τα εκχωρούμε στην ουρά.Ακόμα το πλήθος των αριθμών θα πρέπει να είναι μεγαλύτερο του πλήθους των τελεστών κατά 1 αλλιώς είναι λάθος η postfix.

Στη συνέχεια μέσω του θέματος Α δημιουργούμε ένα αντικείμενο της κλάσης Node<String> οπού το αρχικοποιούμε με το head της ουράς του postfix και αρχίζουμε με μια επανάληψη να φτιάχνουμε την infix συμβολοσειρά ,η διαδικασία είναι ως εξής:Όσο το περιεχόμενο του κόμβου στην επανάληψη δεν είναι τελεστής προχωράμε ,όταν βρούμε τελεστή θα τροποποιήσουμε το περιεχόμενο του κόμβου στον οποίο βρισκόμαστε με την εξής σειρά :παρένθεση(ανοιχτή) , προ προηγούμενο περιεχόμενο κόμβου ,τελεστής(τωρινός),προηγούμενο περιεχόμενο κόμβου ,παρένθεση(κλειστή).Τα προ προηγούμενα και προηγούμενα στοιχεία του κόμβου τα κάνουμε ίσα με κενό("") και εφόσον ο πάρα προ προηγούμενο κόμβος υπάρχει(.previous.previous.previous) το previous του κόμβου μας γίνεται ίσο με αυτό, αλλιώς με null.(Πολυπλοκότητα O(N):Λόγω της μιας επανάληψης).



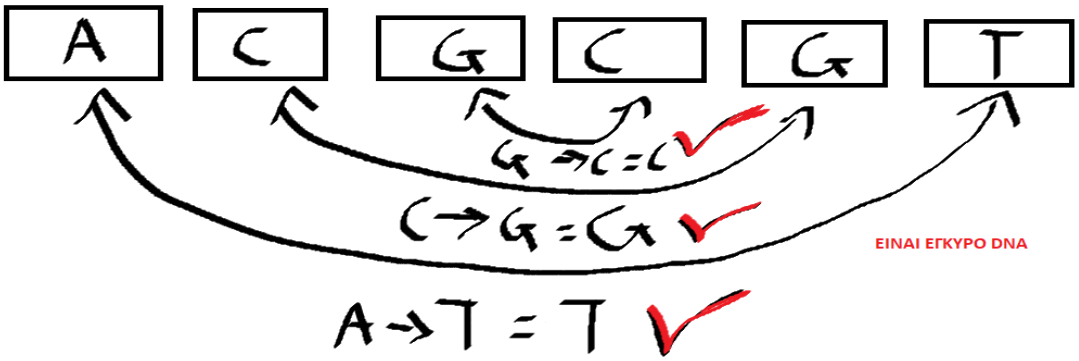
*Παράδειγμα αν δοθεί η postfix 678+**

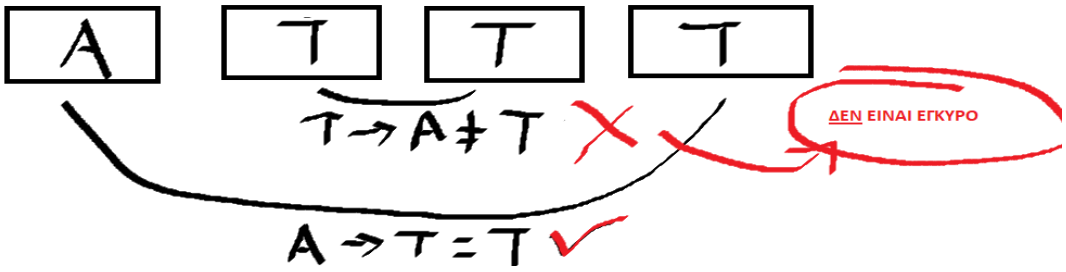
Τέλος ,υλοποιούμε μια μικρή καθυστέρηση 1 sec στο πρόγραμμα μας με τη βοήθεια του Thread.sleep() και εκτυπώνουμε την πολυαναμενόμενη infix συμβολοσειρά με τη βοήθεια της printQueue του θέματος Α.

ΘΕΜΑ Γ

Αρχικά ,διαβάζουμε το input της κλάσης και ελέγχουμε αν το μέγεθος της συμβολοσειράς του DNA είναι άρτιος ,αν δεν είναι βγάζουμε μήνυμα λάθους καθώς δεν γίνεται περιττού μεγέθους DNA να είναι Watson-Crick complemented palindrome.Στη συνέχεια βάζουμε τον κάθε χαρακτήρα στην ουρά αν πληροί τις προϋποθέσεις(να είναι A,T,C ή G) και εισάγουμε μια καθυστέρηση(0.9 sec) για λογούς ομαλότητας(και αισθητικής)του προγράμματος.

Φτάνοντας στο κύριο σημείο του προγράμματος ορίζουμε 2 μεταβλητές τύπου Node<String> οπού θα βρίσκονται το head και το tail της συμβολοσειράς DNA και κάθε φορά που το συμπληρωματικό(T->A C->G και το αντίστροφο) του head θα είναι ίσο με το τελευταίο κόμβο της ουράς η επανάληψη θα συνεχίζει με το head να γίνεται το head.next και το tail να παίρνει την τιμή του tail.previous μέχρι να φτάσουμε στη μέση της ουράς ,να διακοπεί η επανάληψη και να εμφανιστεί το μήνυμα ότι το DNA είναι έγκυρο .Αντίθετα αν κάποιο συμπληρωματικό του head.item δεν ταυτίζεται με την τιμή του tail.item η επανάληψη θα σταματά και θα εκτυπώνεται μήνυμα λάθους στην οθόνη.(Ακολουθεί εικονική αναπαράσταση ενός Watson-Crick και ενός μη).





Υ.Γ. τα προγράμματα γράφθηκαν και έγιναν compile στο Eclipse IDE.