Κολοβος Κωνσταντινος 3170248

Σταγάκης Βασίλειος 3150164

Δομές Δεδομένων Εργασία 1η

Μερος Α

Ορίζουμε μια Κλάση Node που θα αναπαριστα τους κομβους που θα χρησιμοποιηθουν στους ΑΤΔ που θα υλοποιησουμε στη συνεχεια.
Στις διεπαφες που υλοποιουμε η διαδικασια ειναι παρομοια οσον αφορα την χρησιμοποιηση της Node. Εφοσον η Node δεχεται σαν
ορισμα αντικειμενα μπορει να χρησιμοποιηθει και στους δυο ΑΤΔ οπου εχουμε σαν ορισματα int και string αντιστοιχα , αρκει
να κανουμε καποια πολυ απλα cast στον τυπο δεδομενων που χρησιμοποιουμε. Οσον αφορα την υλοποιηση των διεπαφων , ακολουθουμε
τις οδηγιες που μας δινονται , ουσιαστικα δημιουργουμε λιστες μονης συνδεσης και ολες οι μεθοδοι που φτιαχνουμε τρεχουν σε O(1).

Μερος Β

Στο μερος Β παιρνουμε σαν ορισμα του προγραμματος το path του εκαστοτε html αρχειου που θα τρεξουμε. Το ορισμα το παιρνουμε απο
το args[0] ο οποιο δεν ειναι τιποτα αλλο παρα ενα path σε μορφη String. Στο αρχειο TagMatching δημιουργουμε ενα αντικειμενο που παιρνει σαν ορισμα
το path και τρεχουμε τη βασικη μεθοδο της κλασης LoadFile οπου ειναι το μεγαο μερος την υλοποιησης του Β ερωτηματος. Επομενως στο LoadFile πλεον
επιλεγουμε την κλαση BufferedReader για τον χειρισμο των input αρχειων. Το προγραμμα διαβαζει γραμμη προς γραμμη το input και αφου κανει trim
σε καθε γραμμη ψαχνει συγκεριμενα για το συμβολο "<" για να εντοπισει ενα html tag. Στη συνεχεια ελεγχει εαν το αμεσος επομενο συμβολο
ειναι "/" ωστε να διαπιστωθει εαν ειναι tag που ανηγει ή κλεινει. Καθε tag απομονωνεται χρησιμοποιωντας τη μεθοδο substring της κλασης String.
Εδω ερχεται η διεπαφη της στοιβας απο το Α μερος όπου κανουμε push στην στοιβα ενος αντικειμενου της StringStackImpl καθε tag που ανιχνευουμε. Συγκεκριμενα
στην περιπτωση που το tag που ανιχνευουμε ειναι closing tag , χρησιμοποιωντας τη μεθοδο peek ελεγχουμε εαν το τελευταιο στοιχειο που εχουμε εισαγει στη
στοιβα ειναι το opening tag ζευγαρι του. Εαν ειναι το σωστο closing tag εν το εισαγουμε στη στοιβα και διαγραφουμε το opening tag που ειχε εισαχθει πριν απο αυτο.
Εαν δεν ειναι το σωστο closing tag το εισαγουμε στη στοιβα. Με αυτο το τροπο εαν ολα τα tag του αρχειου ειναι σωστα στο τελος η στοιβα μας θα ειναι αδεια , αλλιως
καπου υπαρχει προβλημα.

Μερος Γ

Στο μερος Γ ομοιως με το μερος Β παιρνουμε σαν ορισμα το path του αρχειου με τις αγοραπωλησιες , μεσω args[0]. Στο αρχειο NetBenefit καλουμε τη κλαση ReadTxt οπου υλοποιουμε το μεγαλυτερο μερος του προγραμματος. Το input αρχειο ξεκιναει παντα με buy και τελειωνει παντα με sell. Αρχικά με τη BufferedReader διαβαζουμε τις γραμμες του αρχειου γραμμη προς γραμμη. Η πρωτη λεξη που βρισκουμε ειναι ο τυπος της αγοραπωλησιας(buy or sell) , η δευτερη ο αριθμος των μετοχων και η τεταρτη η τιμη τους. Εαν προκειται για αγορα μετοχών προσθετουμε τον αριθμο τους στο γενικο αθροισμα των μετοχων που διαθετουμε(sumshares) ωστε μελλοντικα να κανουμε ελεγχο για την περιπτωση που οι μετοχες που πουλαμε να ειναι περισσοτερες απο αυτες που διαθετυμε και να τυπωνεται το αντιστοιχο μηνυμα. Στη συνεχεια ο αριθμος των μετοχων που αγοραζουμε και η τιμη με την οποια τις αγοραζουμε μπαινουν με αυτην ακριβως τη σειρα σε ενα αντικειμενο τυπου ουρας απο το Α ερωτημα. Επειδη ειναι ουρα FIFO το πρωτο πραγμα που θα βγαλουμε απο τη δομη θα ειναι ο αριθμος των μετοχων της εκαστοτε αγοραπωλησιας. Με αυτον τον τροπο οταν βρουμε μια γραμμη στο αρχειο οπου εχουμε πωληση μετοχων , θα βγαλουμε απο την ουρα πρωτα το ζευγος μετοχων-τιμης που ειχε εισαχθει πρωτο , δηλαδη την παλαιοτερη αγορα. Καθε φορα που βγαινει μια αγορα απο την ουρα , ελεγχουμε εαν οι μετοχες που θελουμε να πουλησουμε ειναι περισσοτερες απο αυτες της αγορας. Εαν οχι τοτε πουλαμε οσες χρειαζομαστε. Εαν ναι τοτε πουλαμε ολες της μετοχες την πρωτης αογρας και παμε στην επομενη μεχρι να φτασουμε τον αριθμο που θελουμε να πουλησουμε.
Παντα αμεσσως μετα τον αριθμο τον μετοχων βγαζουμε απο την ουρα την αντιστοιχη τιμη και υπολογιζουμε την τιμη κερδους/ζημιας οπως περιγραφεται στην εκφωνηση της εργασιας.