Προγραμματιστική Εργασία - 2ο Μέρος
ΗΥ-240: Δομές Δεδομένων
Εαρινό Εξάμηνο - Ακαδημαϊκό Έτος 2020
Διδάσκουσα: Παναγιώτα Φατούρου

Ημερομηνία Παράδοσης: Δευτέρα, 18 Μαΐου 2020

Τρόπος Παράδοσης: Χρησιμοποιώντας το πρόγραμμα turnin. Πληροφορίες για το πώς λειτουργεί το πρόγραμμα turnin παρέχονται στην ιστοσελίδα του μαθήματος.

Γενική Περιγραφή: Στην εργασία αυτή καλείστε να υλοποιήσετε ένα πρόγραμμα που να προσομοιώνει τη λειτουργία ενός ομίλου αεροπορικών εταιρειών που εκτελούν πτήσεις σε διάφορες χώρες. Το πρόγραμμα που θα υλοποιήσετε θα πρέπει να επιτρέπει στον όμιλο να συμπεριλαμβάνει καινούριες εταιρείες (companies). Η κάθε εταιρεία έχει ένα στόλο από αεροπλάνα (airplanes), καθένα εκ των οποίων εκτελεί πτήσεις προς ένα συγκεκριμένο προορισμό. Ο όμιλος παρέχει πτήσεις προς συγκεκριμένους προορισμούς (destinations) μέσω των διαφορετικών εταιρειών που τον απαρτίζουν.

Νέο: Ο όμιλος καταγράφει τους πελάτες (clients) που χρησιμοποιούν κάποια από τις εταιρείες για τα ταξίδια τους μαζί με τα μίλια που συγκεντρώνουν.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Το σύστημα αποτελείται από ένα σύνολο εταιρειών που απαρτίζουν τον όμιλο και αποθηκεύονται σε μία **κυκλική, διπλά συνδεδεμένη, ταξινομημένη λίστα** (με εικονικό-dummy κόμβο) που ονομάζεται **λίστα αεροπορικών εταιρειών (airlines)**. Οι εταιρείες είναι ταξινομημένες με βάση το αναγνωριστικό τους, σε αύξουσα διάταξη.

Ο κάθε κόμβος της λίστας αντιστοιχεί σε μια αεροπορική εταιρεία και υλοποιείται ως μια εγγραφή τύπου struct airline με τα ακόλουθα πεδία:

- **aid:** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά την αεροπορική εταιρεία.
- **airplane:** Δείκτης (τύπου struct airplane *) στη ρίζα ενός **δυαδικού δένδρου αναζήτησης** που ονομάζεται **δένδρο αεροπλάνων της αεροπορικής εταιρείας** με αναγνωριστικό aid.
- **next:** Δείκτης (τύπου struct airline *) στον επόμενο κόμβο της λίστας αεροπορικών εταιρειών.
- **prev:** Δείκτης (τύπου struct airline *) στον προηγούμενο κόμβο της λίστας αεροπορικών εταιρειών.

Το δένδρο αεροπλάνων μιας αεροπορικής εταιρείας είναι ταξινομημένο με βάση το αναγνωριστικό του κάθε αεροπλάνου (είναι επομένως δυαδικό δένδρο αναζήτησης). Κάθε στοιχείο του δένδρου αντιστοιχεί σε ένα αεροπλάνο της αεροπορικής εταιρείας και είναι μια εγγραφή τύπου struct airplane με τα παρακάτω πεδία:

- **pid:** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το αεροπλάνο.
- **dest:** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον προορισμό στον οποίο ταξιδεύει το αεροπλάνο. Το εύρος τιμών αυτού του πεδίου είναι μεταξύ [0, 9].
- **depart_time:** Αναγνωριστικό (τύπου int) που αντιστοιχεί στην ώρα αναχώρησης του αεροπλάνου προς τον προορισμό του. Η ώρα θα έχει τη μορφή HHMM. Για παράδειγμα, η ώρα 1620 αντιστοιχεί στην ώρα 16:20. Υποθέτουμε, για λόγους απλούστευσης της εργασίας, ότι κάθε αεροπλάνο πετάει καθημερινά μία φορά προς το μοναδικό προορισμό του.
- **lc:** Δείκτης (τύπου struct airplane *) στον αριστερό θυγατρικό κόμβο του κόμβου στον οποίο αντιστοιχεί το αεροπλάνο με αναγνωριστικό pid.
- **rc:** Δείκτης (τύπου struct airplane *) στο δεξιό θυγατρικό κόμβο του κόμβου στον οποίο αντιστοιχεί το αεροπλάνο με αναγνωριστικό pid.

Στο **Σχήμα 1** (σελ 5) παρουσιάζεται η λίστα των αεροπορικών εταιρειών, ο κάθε κόμβος της οποίας περιέχει ένα δείκτη προς τη ρίζα του δένδρου αεροπλάνων της αντίστοιχης αεροπορικής εταιρείας.

Το σύστημα περιέχει επίσης ένα μηχανισμό για την κατηγοριοποίηση των αεροπλάνων στους προορισμούς που ταξιδεύουν. Ο μηχανισμός αυτός υλοποιείται με ένα πίνακα **DESTINATIONS** που ονομάζεται **πίνακας προορισμών**. Θεωρούμε ότι υπάρχουν συνολικά 10 προορισμοί και έτσι ο πίνακας έχει 10 θέσεις, μια για κάθε προορισμό.

Κάθε στοιχείο i του πίνακα περιέχει ένα δείκτη στη ρίζα ενός **ταξινομημένου, δυαδικού, νηματικού δένδρου** (δυαδικό νηματικό δένδρο αναζήτησης) που ονομάζεται **δένδρο πτήσεων του προορισμού i** και περιέχει όλα τα αεροπλάνα που ταξιδεύουν προς τον προορισμό i. Το δένδρο αυτό είναι ταξινομημένο με βάση την ώρα αναχώρησης των πτήσεων. Σε περίπτωση που δύο πτήσεις αντιστοιχούν στην ίδια ώρα, η σειρά εμφάνισής τους στο δένδρο θα είναι με βάση το αναγνωριστικό του αεροπλάνου.

Ο κάθε κόμβος v του δένδρου είναι μια εγγραφή τύπου struct flight με τα ακόλουθα πεδία:

- **pid:** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το αεροπλάνο (και τον κόμβο v).
- **depart_time:** Αναγνωριστικό (τύπου int) που αντιστοιχεί στην ώρα αναχώρησης του αεροπλάνου προς τον προορισμό του. Η ώρα θα έχει τη μορφή HHMM. Για παράδειγμα, η ώρα 1620 αντιστοιχεί στην ώρα 16:20. Υποθέτουμε, για λόγους απλούστευσης της εργασίας, ότι κάθε αεροπλάνο πετάει καθημερινά μία φορά προς το μοναδικό προορισμό του.
- **lc:** Δείκτης (τύπου struct flight *) στον αριστερό θυγατρικό κόμβο του κόμβου v.
- **rc:** Δείκτης (τύπου struct flight *) στο δεξίο θυγατρικό κόμβο του κόμβου v.
- **thread_status:** Πεδίο (τύπου int) που χαρακτηρίζει τους lc και rc δείκτες του κόμβου v, είτε ως νηματικούς είτε όχι. Το εύρος τιμών αυτού του πεδίου είναι [0, 3].
  - Ένας κόμβος v με δύο κανονικά παιδιά, έχει **thread_status = 0**.
  - Αν ο v δεν έχει αριστερό παιδί, έχει **thread_status = 1**, το οποίο υποδηλώνει ότι ο lc δείκτης του είναι νηματικός και άρα δείχνει στον προηγούμενο του κόμβο v στο δένδρο σύμφωνα με την ενδοδιατεταγμένη διάσχιση (inorder predecessor).
  - Αν ο v δεν έχει δεξί παιδί, έχει **thread_status = 2**, το οποίο υποδηλώνει ότι ο rc δείκτης του είναι νηματικός και δείχνει στον επόμενο του κόμβο v σύμφωνα με την ενδοδιατεταγμένη διάσχιση (inorder successor).
  - Αν ο v δεν έχει κανένα παιδί, έχει **thread_status = 3**, το οποίο υποδηλώνει ότι οι lc, rc δείκτες του είναι νηματικοί και δείχνουν στον προηγούμενο και στον επόμενο του v, αντίστοιχα, σύμφωνα με την ενδοδιατεταγμένη διάσχιση.

Στο **Σχήμα 2** (σελ 6) παρουσιάζεται ο πίνακας προορισμών, κάθε στοιχείο του οποίου περιέχει έναν δείκτη προς τη ρίζα του δένδρου πτήσεων του προορισμού που αντιστοιχεί σ' αυτό το στοιχείο του πίνακα.

Το σύστημα επίσης διατηρεί μία **ουρά προτεραιότητας** με όλους τους πελάτες του ομίλου για την απόδοση μιλίων σύμφωνα με τις πτήσεις που πραγματοποιούν. Η ουρά αυτή υλοποιείται με ένα **μερικώς διατεταγμένο (partially ordered), τριαδικό (συνδεδεμένο) δένδρο** και ονομάζεται **ουρά προτεραιότητας πελατών (clients)**. (Επομένως, στην εργασία αυτή, η ουρά προτεραιότητας δεν υλοποιείται ως σωρός.) Η ταξινόμηση των πελατών γίνεται με βάση τα μίλια που αυτοί έχουν συγκεντρώσει, ώστε ο πελάτης με τα περισσότερα μίλια να έχει την μέγιστη προτεραιότητα (και άρα να βρίσκεται στη ρίζα). Η ουρά προτεραιότητας είναι επομένως μια ουρά προτεραιότητας μεγίστου (δηλαδή πελάτες με περισσότερα μίλια θεωρούνται μεγαλύτερης προτεραιότητας). Οι πελάτες μαζεύουν μίλια με κάθε ταξίδι που

πραγματοποιούν, ανάλογα με τον προορισμό της πτήσης τους. Συγκεκριμένα, κάθε προορισμός αποδίδει σε κάθε πελάτη που ταξιδεύει προς αυτόν τον προορισμό, 100 μίλια επί τη θέση του (index + 1) στον πίνακα DESTINATIONS, με στόχο την απόδοση περισσότερων μιλίων στους μακρινότερους προορισμούς. Για παράδειγμα, ένας κοντινός προορισμός που βρίσκεται στη 2η θέση του πίνακα αποδίδει 200 μίλια σε κάθε ταξίδι.

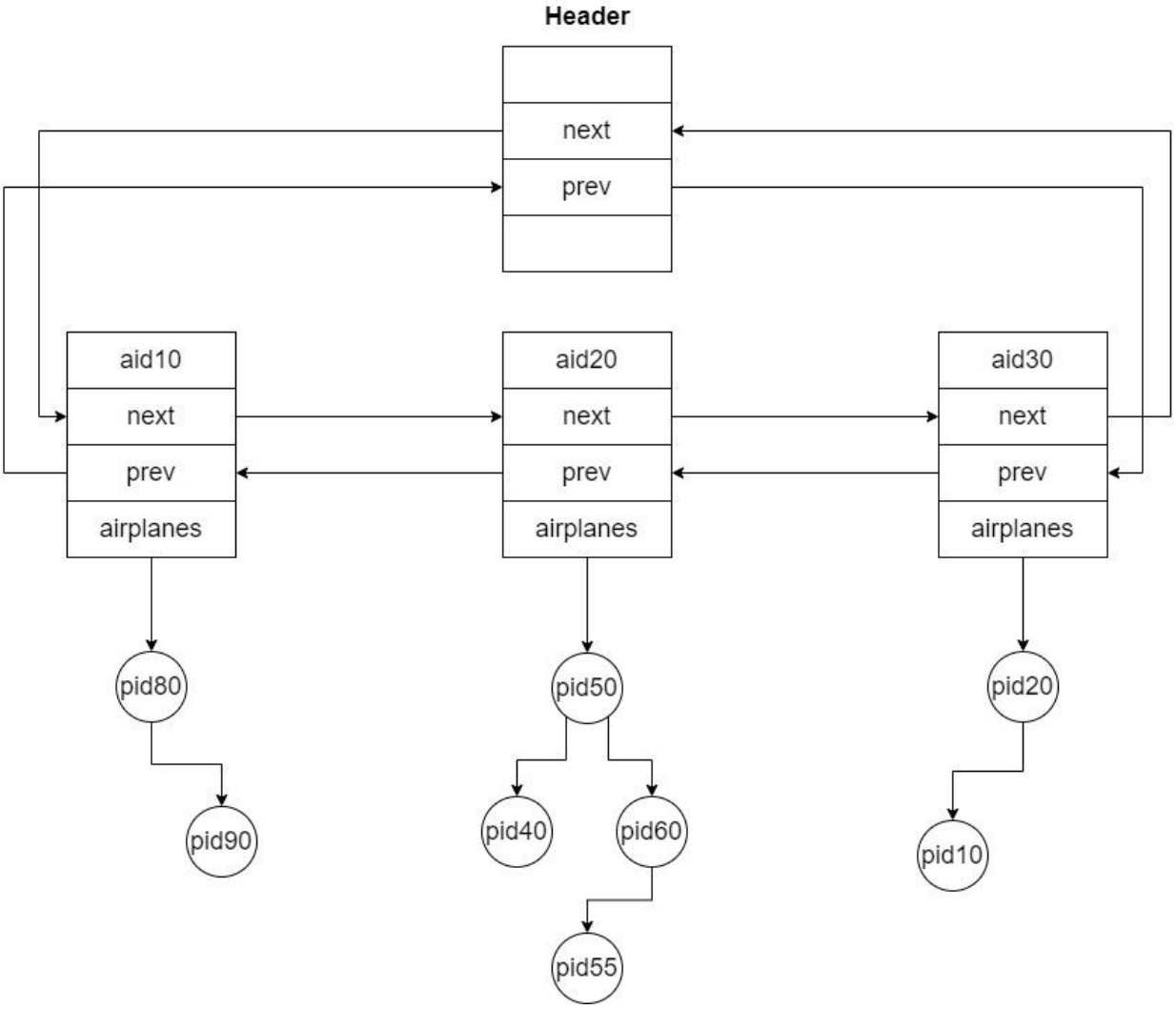
Ο κάθε κόμβος του πλήρους (συνδεδεμένου) δένδρου που υλοποιεί την ουρά προτεραιότητας είναι μια εγγραφή τύπου struct client με τα ακόλουθα πεδία:

- **cid:** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον πελάτη (και τον κόμβο που του αντιστοιχεί).
- **miles:** Αναγνωριστικό (τύπου int) που αναπαριστά τα μίλια που αντιστοιχούν στον πελάτη.
- **lc:** Δείκτης (τύπου struct client *) στον αριστερό θυγατρικό κόμβο του κόμβου με αναγνωριστκό cid.
- **mc:** Δείκτης (τύπου struct client *) στο μεσαίο θυγατρικό κόμβο του κόμβου με αναγνωριστικό cid.
- **rc:** Δείκτης (τύπου struct client *) στο δεξιό θυγατρικό κόμβο του κόμβου με αναγνωριστικό cid.

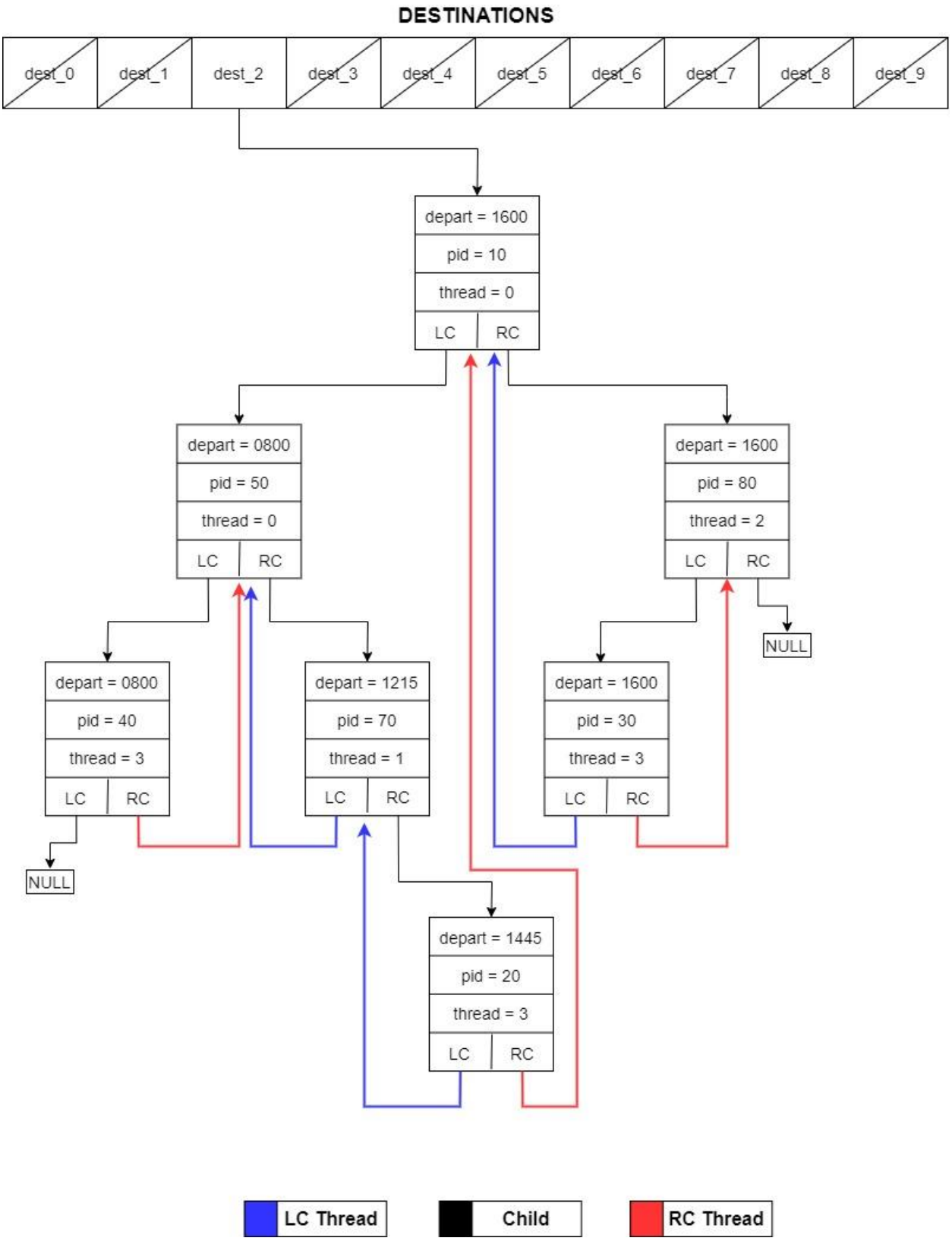
Στο **Σχήμα 3** (σελ 8) παρουσιάζεται το δένδρο πελατών.

Επίσης υπάρχει ένα struct τύπου priority_queue που αποθηκεύει ένα δείκτη στη ρίζα του πλήρους δένδρου που υλοποιεί την ουρά προτεραιότητας και έναν μετρητή που αποθηκεύει πόσα στοιχεία περιέχει η ουρά προτεραιότητας την εκάστοτε χρονική στιγμή:

- struct client *Q: δείκτης στη ρίζα του πλήρους δένδρου που υλοποιεί την ουρά προτεραιότητας.
- int num_elements: μετρητής που αποθηκεύει το πλήθος των στοιχείων της ουράς προτεραιότητας κάθε χρονική στιγμή.

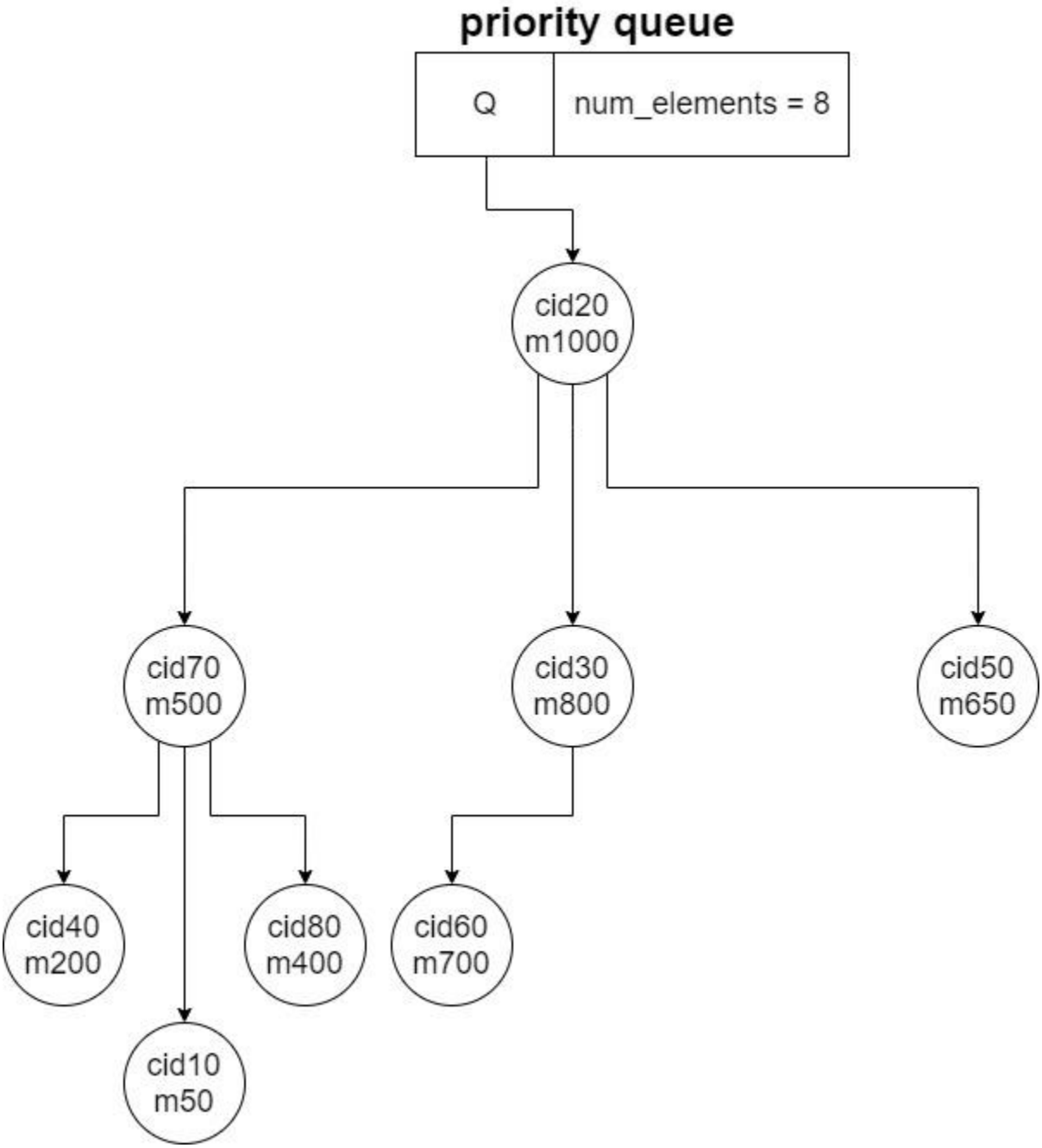


Σχήμα 1: Η διπλά συνδεδεμένη λίστα αεροπορικών εταιρειών, η οποία είναι ταξινομημένη με βάση το αναγνωριστικό (aid) των κόμβων της. Επίσης, το δυαδικό δένδρο αναζήτησης αεροπλάνων για κάθε εταιρεία, ταξινομημένο με βάση το αναγνωριστικό του κάθε αεροπλάνου (pid).



Σχήμα 2: Ο πίνακας των προορισμών, κάθε στοιχείο του οποίου αποθηκεύει ένα δείκτη στο

νηματικό δέντρο πτήσεων του προορισμού. Τα δέντρα πτήσεων είναι ταξινομημένα βάσει της ώρας αναχώρησης (depart) κι όταν η ώρα αναχώρησης είναι ίδια μεταξύ δύο ή περισσότερων κόμβων, βάσει του αναγνωριστικού αεροπλάνου που εκτελεί την πτήση (pid). Κάθε δείκτης που απεικονίζεται με μαύρο χρώμα αναπαριστά είτε έναν lc ή έναν rc δείκτη ενός κόμβου προς τον αντίστοιχο θυγατρικό κόμβο του κόμβου από τον οποίο εκκινεί, κάθε δείκτης που απεικονίζεται με μπλε χρώμα είναι νηματικός και δείχνει στον προηγούμενου του κόμβου από τον οποίο εκκινεί, σύμφωνα με την ενδοδιατεταγμένη διάσχιση (inorder predecessor), ενώ κάθε δείκτης που απεικονίζεται με κόκκινο χρώμα είναι επίσης νηματικός και δείχνει στον επόμενο του κόμβου από τον οποίο εκκινεί, σύμφωνα με την ενδοδιατεταγμένη διάσχιση (inorder successor). Το είδος των δεικτών κάθε κόμβου χαρακτηρίζεται από το πεδίο thread_status (thread) (το οποίο παίρνει τιμές στο διάστημα [0, 4], όπως περιγράφηκε παραπάνω).



Σχήμα 3: Η ουρά προτεραιότητας πελατών, στην οποίο κάθε κόμβος περιέχει τα στοιχεία ενός πελάτη με αναγνωριστικό cid και μίλια m. Ο πελάτης στη ρίζα του δένδρου έχει τα περισσότερα μίλια και άρα τη μέγιστη προτεραιότητα. Το δένδρο που αναπαριστά την ουρά προτεραιότητας είναι πλήρες.

Τρόπος Λειτουργίας Προγράμματος

Το πρόγραμμα που θα δημιουργηθεί θα πρέπει να εκτελείται καλώντας την ακόλουθη εντολή:

<executable> <input-file>

όπου <executable> είναι το όνομα του εκτελέσιμου αρχείου του προγράμματος (π.χ. a.out) και <input-file> είναι το όνομα ενός αρχείου εισόδου (π.χ. testfile) το οποίο περιέχει γεγονότα των μορφών που περιγράφονται στη συνέχεια.

R <aid>

Γεγονός τύπου **Register Airline** που υποδηλώνει την εισαγωγή μιας αεροπορικής εταιρείας με αναγνωριστικό <aid> στον όμιλο. Κατά το γεγονός αυτό θα γίνεται εισαγωγή ενός νέου κόμβου τύπου struct airline στη λίστα των αεροπορικών εταιρειών. Μετά την εισαγωγή, η λίστα πρέπει να παραμένει ταξινομημένη. Μετά την εκτέλεση ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

R <aid>

Airlines = $<aid_1>$, $<aid_2>$,...,$<aid_n>$

DONE

όπου n είναι ο αριθμός των κόμβων στη λίστα αεροπορικών εταιρειών και για κάθε $i \in \{1, \ldots, n\}$, $<aid_i>$ είναι το αναγνωριστικό της αεροπορικής εταιρείας που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

I <pid> <aid> <dest> <depart_time>

Γεγονός τύπου **Insert Airplane** που υποδηλώνει την εισαγωγή ενός αεροπλάνου με αναγνωριστικό <pid> στο δένδρο αεροπλάνων της εταιρείας με αναγνωριστικό <aid>. Το αεροπλάνο πετάει προς τον προορισμό με αναγνωριστικό <dest> και η ώρα αναχώρησης του είναι η <depart_time>.

Κατά το γεγονός αυτό θα πρέπει να πραγματοποιούνται οι ακόλουθες ενέργειες. Αρχικά, θα αναζητάτε στη λίστα των αεροπορικών εταιρειών την εταιρεία με αναγνωριστικό <aid> και στη συνέχεια θα εισάγετε στο δένδρο των αεροπλάνων της αεροπορικής εταιρείας ένα νέο κόμβο με αναγνωριστικό <pid>. Μετά από κάθε εισαγωγή, το δένδρο των αεροπλάνων θα πρέπει να παραμένει ταξινομημένο.

Τέλος, θα πρέπει να εισάγετε ένα νέο κόμβο στον πίνακα προορισμών. Για την εισαγωγή αυτή, θα πρέπει να χρησιμοποιήσετε την παράμετρο <dest> του γεγονότος για να εντοπίσετε το κατάλληλο δένδρο στον πίνακα προορισμών και έπειτα θα εισάγετε ένα νέο κόμβο που να αντιπροσωπεύει τη νέα πτήση. Το δένδρο πτήσεων θα πρέπει να παραμένει ταξινομημένο μετά το πέρας της εισαγωγής, ενώ θα πρέπει να κάνετε ότι αλλαγές χρειάζονται στα νήματα

για την διατήρηση της σωστής λειτουργίας τους.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

I <pid> <aid> <dest> <depart_time>

Airline <aid> = <pid_1 : aid_1>, <pid_2 : aid_2>, …, <pid_m : aid_m>
Destination$_{<dest>}$ = <pid'_1 : t_1>, <pid_2 : t_2>, ..., <pid'_k : t_k>

DONE

όπου m είναι το πλήθος των κόμβων της λίστας των αεροπλάνων της αεροπορικής εταιρείας με αναγνωριστικό <aid> και για κάθε i, $1 \le i \le m$, pid_i είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχεί στον i-οστό κόμβο του δένδρου των αεροπλάνων της εταιρείας, σύμφωνα με την ενδοδιατεταγμένη διάσχιση. Επίσης, για κάθε j, $1 \le j \le k$, όπου k είναι ο αριθμός πτήσεων στο δέντρο πτήσεων του προορισμού <dest>, pid_j είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχεί στον f-οστό κόμβο του δένδρου πτήσεων, σύμφωνα με την ενδοδιατεταγμένη διάσχιση και t_j είναι η ώρα αναχώρησής του.

C <aid> <pid> <dest>

Γεγονός τύπου **Cancel Flight** που υποδηλώνει την απόσυρση του αεροπλάνου με αναγνωριστικό <pid> από το στόλο της αεροπορικής εταιρείας με αναγνωριστικό <aid>.

Κατά το γεγονός αυτό θα πρέπει να αναζητήσετε την εταιρεία με αναγνωριστικό <aid> στη λίστα αεροπορικών εταιρειών και στη συνέχεια να αφαιρέσετε από το δένδρο αεροπλάνων της το αεροπλάνο με αναγνωριστικό <pid>. Το δένδρο αεροπλάνων πρέπει να παραμείνει ταξινομημένο μετά τη διαγραφή.

Επιπρόσθετα, θα πρέπει να χρησιμοποιήσετε την παράμετρο <dest> του γεγονότος για να εντοπίσετε το κατάλληλο δένδρο στον πίνακα προορισμών και έπειτα θα πρέπει να αφαιρέσετε τον κόμβο με αναγνωριστικό <pid> που αντιπροσωπεύει το αεροπλάνο προς απόσυρση. Το δένδρο πτήσεων θα πρέπει να παραμείνει ταξινομημένο μετά τη διαγραφή, ενώ θα πρέπει να κάνετε ότι αλλαγές χρειάζονται στα νήματα για την διατήρηση της σωστής λειτουργίας τους.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

C <aid> <pid> <destination>

Airline <aid> = <pid_1 : aid_1>, <pid_2 >, …, <pid_m >
Destination$_{<dest>}$ = <pid'_1 : t_1>, <pid_2 : t_2>, ..., <pid'_k : t_k>

```
DONE
```

όπου m είναι το πλήθος των κόμβων της λίστας των αεροπλάνων της αεροπορικής εταιρείας με αναγνωριστικό <aid> και για κάθε i, $1 \le i \le m$, pid_i είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχεί στον i-οστό κόμβο του δένδρου των αεροπλάνων της εταιρείας, σύμφωνα με την ενδοδιατεταγμένη διάσχιση. Επίσης, για κάθε j, $1 \le j \le k$, όπου k είναι ο αριθμός πτήσεων στο δέντρο πτήσεων του προορισμού <dest>, pid_j είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχεί στον f-οστό κόμβο του δένδρου πτήσεων, σύμφωνα με την ενδοδιατεταγμένη διάσχιση και t_j είναι η ώρα αναχώρησής του.

D <aid>

Γεγονός τύπου **Delete Airline** που υποδηλώνει τη διαγραφή της εταιρείας με αναγνωριστικό <aid> από τον όμιλο.

Κατά το γεγονός αυτό, θα πρέπει να αναζητήσετε στη λίστα αεροπορικών εταιρειών την εταιρεία με αναγνωριστικό <aid> και στη συνέχεια να διατρέξετε και να αποσύρετε όλα τα αεροπλάνα από το δένδρο αεροπλάνων της.

Για κάθε αεροπλάνο που βρίσκεται στο δένδρο αεροπλάνων της εταιρείας θα πρέπει να διαγράφετε την πτήση που του αντιστοιχεί από τον σωστό δένδρο του πίνακα προορισμών. Αυτό το δένδρο πτήσεων θα πρέπει να παραμείνει ταξινομημένο μετά τη διαγραφή, ενώ θα πρέπει να κάνετε ότι αλλαγές χρειάζονται στα νήματα για την διατήρηση της σωστής λειτουργίας τους.

Τέλος, θα πρέπει να διαγράψετε από τη λίστα αεροπορικών εταιρειών την εταιρεία με αναγνωριστικό <aid>. Η λίστα θα πρέπει να παραμείνει ταξινομημένη μετά τη διαγραφή.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
D <aid>
    Airlines
        Airline <aid_1> = <pid_1,1>, <pid_1,2>, ..., <pid_1,m1 >
        Airline <aid_2> = <pid_2,1>, <pid_2,2>, ..., <pid_2,m2 >
        ...
        Airline  <aid_n> = <pid_n,1>, <pid_n,2>, ..., <pid_n,mn >
    Destinations
        Destination_1 = <pid_1,1 : t_1,1>, ..., <pid_1,k1 : t_1,k1>
        …
        Destination_10 = <pid_10,1 : t_10,1>, ..., <pid_10,k10 : t_10,k10>
DONE
```

όπου n είναι το πλήθος των αεροπορικών εταιρειών στη λίστα αεροπορικών εταιρειών, για

κάθε i, $1 \le i \le n$, aid_i είναι το αναγνωριστικό του i-οστού κόμβου στη λίστα των αεροπορικών εταιρειών, m_i είναι το πλήθος των κόμβων της λίστας των αεροπλάνων της i-οστής εταιρείας στη λίστα και για κάθε $j \in \{1, ..., m_i\}$, $pid_{i,j}$ είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχεί στον j-οστό κόμβο του δένδρου των αεροπλάνων της i-οστής εταιρείας, σύμφωνα με την ενδοδιατεταγμένη διάσχιση. Επίσης, για κάθε f, $1 \le f \le 10$, όπου k_f είναι ο αριθμός πτήσεων στο δέντρο πτήσεων του προορισμού $dest_f$ και για κάθε l, $1 \le l \le k_f$, $pid_{f,l}$ είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχούν στον f-οστό κόμβο του δένδρου πτήσεων, σύμφωνα με την ενδοδιατεταγμένη διάσχιση και t_f είναι η ώρα αναχώρησής του.

A <aid1> <aid2>

Γεγονός τύπου **Acquisition Airline** το οποίο σηματοδοτεί την εξαγορά της αεροπορικής εταιρείας <aid1> από την αεροπορική εταιρεία <aid2>.

Κατά το γεγονός αυτό, θα πρέπει να αναζητήσετε στη λίστα των αεροπορικών εταιρειών τις εταιρείες με αναγνωριστικά <aid1> και <aid2>, αντίστοιχα και να συνενώσετε τα δένδρα αεροπλάνων των εταιρειών αυτών. Το δένδρο αεροπλάνων της αεροπορικής εταιρείας με αναγνωριστικό <aid2> που θα προκύψει μετά τη συνένωση θα πρέπει να είναι ταξινομημένο, ενώ το δένδρο αεροπλάνων της αεροπορικής εταιρείας <aid1> θα πρέπει να είναι κενό. Η χρονική πολυπλοκότητα αυτού του γεγονότος θα πρέπει να είναι O(n1+n2), όπου n1 και n2 είναι το πλήθος στοιχείων των δένδρων αεροπλάνων των αεροπορικών εταιρειών <aid1> και <aid2>, αντίστοιχα. Τέλος θα πρέπει να διαγράψετε την εταιρεία με αναγνωριστικό <aid1> από τη λίστα των αεροπορικών εταιρειών.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

A <aid1> <aid2>

Airline $<aid_1>$ = $<pid_{1,1}>$, $<pid_{1,2}>$, …, $<pid_{1,m1}>$
Airline $<aid_2>$ = $<pid_{2,1}>$, $<pid_{2,2}>$, …, $<pid_{2,m2}>$
...
Airline $<aid_n>$ = $<pid_{n,1}>$, $<pid_{n,2}>$, ..., $<pid_{n,mn}>$

DONE

όπου n είναι το πλήθος των αεροπορικών εταιρειών στη λίστα αεροπορικών εταιρειών, για κάθε i, $1 \le i \le n$, aid_i είναι το αναγνωριστικό του i-οστού κόμβου στη λίστα των αεροπορικών εταιρειών, m_i είναι το πλήθος των κόμβων της λίστας των αεροπλάνων της i-οστής εταιρείας και για κάθε $j \in \{1, ..., m_i\}$, $pid_{i,j}$ είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχεί στον j-οστό κόμβο του δένδρου των αεροπλάνων της i-οστής εταιρείας, σύμφωνα με την ενδοδιατεταγμένη διάσχιση.

S <aid1> <aid2> <dest>

Γεγονός τύπου **Subsidiary Airline**, όπου η εταιρεία με αναγνωριστικό <aid2> εξαγοράζει τα αεροπλάνα της αεροπορικής εταιρείας με αναγνωριστικό <aid1>, τα οποία εκτελούν πτήση

προς ένα συγκεκριμένο προορισμό.

Κατά το γεγονός αυτό, θα πρέπει να αναζητήσετε στη λίστα των αεροπορικών εταιρειών τις εταιρείες με αναγνωριστικά <aid1> και <aid2>. Έπειτα, θα πρέπει να διατρέξετε το δένδρο των αεροπλάνων της <aid1> και για κάθε αεροπλάνο που εκτελεί πτήση προς τον προορισμό <dest>, θα πρέπει να το διαγράφετε από αυτό το δένδρο και να το εισάγετε στο δένδρο των αεροπλάνων της αεροπορικής εταιρείας με αναγνωριστικό <aid2>. Το δένδρο των αεροπλάνων της εταιρείας με αναγνωριστικό <aid2> που θα προκύψει μετά την εκτέλεση του γεγονότος αυτού θα πρέπει να είναι ταξινομημένο. Η χρονική πολυπλοκότητα του γεγονότος θα πρέπει να είναι O(m*h), όπου m είναι ο συνολικός αριθμός των αεροπλάνων της εταιρίας της οποίας κάποια αεροπλάνα εξαγοράζονται και h είναι το ύψος του δένδρου των αεροπλάνων της εταιρίας που εξαγοράζει τα αεροπλάνα.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
S <aid1> <aid2> <dest>
        Airline <aid1> = <pid1,1>, <pid1,2>, …, <pid1,m1 >
        Airline <aid2> = <pid2,1>, <pid2,2>, …, <pid2,m2 >
DONE
```

όπου m_1 και m_2 είναι το πλήθος των αεροπορικών εταιρειών στις λίστες αεροπορικών εταιρειών, με αναγνωριστικά <aid1> και <aid2>, αντίστοιχα και για κάθε i, $1 \leq i \leq 2$, και κάθε j $\in \{1, ..., mi\}$, $pid_{i,j}$ είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχεί στον j-οστό κόμβο του δένδρου των αεροπλάνων της εταιρείας με αναγνωριστικό <aid >, σύμφωνα με την ενδοδιατεταγμένη διάσχιση.

F <dest> <ts>

Γεγονός τύπου **Find Flight** όπου ένας πελάτης ψάχνει όλες τις πτήσεις που αναχωρούν ακριβώς μετά την ώρα <ts> με προορισμό <dest>.

Κατά το γεγονός αυτό θα πρέπει να διατρέξετε μέρος του δένδρου πτήσεων του προορισμού <dest> , αναζητώντας αρχικά τον κόμβο με χρονοσφραγίδα <ts> και στη συνέχεια με τη χρήση των νημάτων να διατρέξετε και να εκτυπώσετε τις πτήσεις που αναχωρούν μετά την ώρα <ts> (που έχουν δηλαδή depart_time > ts).

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
F  <dest> <ts>
        Destination_d = <pid_1 : t_1>, ..., <pid_d : t_d>
DONE
```

όπου d είναι το πλήθος των κόμβων στο δένδρο των πτήσεων του προορισμού <dest> και για κάθε i, $1 \le i \le d$, pid_i είναι το αναγνωριστικό του αεροπλάνου που αποθηκεύεται σ΄ αυτόν τον κόμβο και t_i είναι η ώρα αναχώρησής του.

N <cid>

Γεγονός τύπου **New Client**, όπου ένας νέος χρήστης με αναγνωριστικό <cid> εγγράφεται στο σύστημα ως πελάτης του ομίλου.

Κατά το γεγονός αυτό θα γίνεται εισαγωγή ενός νέου κόμβου τύπου struct client στο δένδρο πελατών με μηδενικά μίλια. Συγκεκριμένα, θα χρησιμοποιείτε το πεδίο num_elements του struct priority-queue για να βρείτε τον κόμβο όπου θα πρέπει να τοποθετήσετε τον νέο κόμβο χωρίς να καταστρατηγηθεί η ιδιότητα ότι το δένδρο είναι πλήρες. Μετά την εισαγωγή, ίσως να χρειάζεται να επανορθώσετε την ιδιότητα της μερικής διάταξης στο δένδρο που υλοποιεί την ουρά προτεραιότητας, ανταλλάσσοντας τον κόμβο που μόλις εισαγάγατε με τον γονικό του κόμβο. Η διαδικασία αυτή μπορεί να πρέπει να επαναληφθεί μέχρι ο νέος κόμβος να βρεθεί στη θέση που πρέπει (σύμφωνα με τα μίλια του) ώστε το δένδρο να είναι μερικώς διατεταγμένο. Επομένως, μετά την εισαγωγή, το δέντρο πρέπει να εξακολουθεί να αναπαριστά ουρά προτεραιότητας μεγίστου (με την προτεραιότητα να καθορίζεται βάσει των μιλίων). Η χρονική πολυπλοκότητα του γεγονότος αυτού πρέπει να είναι O(h), όπου h είναι το ύψος του δένδρου που αναπαριστά την ουρά προτεραιότητας.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

N <cid>
Priority Queue = $<cid_1 : miles_1>, <cid_2 : miles_2>, \ldots, <cid_p : miles_p>$
DONE

όπου p είναι το πλήθος στοιχείων στην ουρά προτεραιότητας και για κάθε i, $1 \le i \le p$, cid_i και $miles_i$, είναι το αναγνωριστικό και τα μίλια, αντίστοιχα, του i-οστού πελάτη, **σύμφωνα με διάσχιση κατά επίπεδα** (οπότε θα πρέπει να υλοποιήσετε διάσχιση κατά επίπεδα: δείτε και το γεγονός τύπου Z στη συνέχεια).

E

Γεγονός τύπου **Erase Client** όπου ο πελάτης με τη μεγαλύτερη προτεραιότητα διαγράφεται από το σύστημα.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

E <cid>
Priority Queue = $<cid_1 : miles_1>, <cid_2 : miles_2>, \ldots, <cid_p : miles_p>$

```
DONE
```

όπου p είναι το πλήθος στοιχείων στην ουρά προτεραιότητας και για κάθε i, $1 \le i \le p$, cid_i και $miles_i$, είναι το αναγνωριστικό και τα μίλια, αντίστοιχα, του i-οστού πελάτη, **σύμφωνα με διάσχιση κατά επίπεδα**.

T <cid> <dest>

Γεγονός τύπου **Travel Client** όπου ένας πελάτης με αναγνωριστικό <cid> ταξιδεύει στον προορισμό με αναγνωριστικό <dest>.

Κατά το γεγονός αυτό, στον πελάτη με αναγνωριστικό <cid> θα πρέπει να απονέμονται τα μίλια που αντιστοιχούν στον προορισμό <dest>. Πρώτα θα πρέπει να αναζητείτε τον πελάτη στο δένδρο πελατών (εκτελώντας διάσχιση) και αφού τον βρείτε, θα πρέπει να αυξάνετε τα μίλια του ανάλογα. Με την απονομή των μιλίων φροντίστε να κάνετε ό,τι αλλαγές χρειάζονται ώστε το δένδρο να συνεχίσει να αναπαριστά ουρά προτεραιότητας μεγίστου.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
T <cid> <dest>
        Priority Queue = <cid_1 : miles_1>, <cid_2 : miles_2>, …, <cid_p : miles_p>
DONE
```

όπου p είναι το πλήθος στοιχείων στην ουρά προτεραιότητας και για κάθε i, $1 \le i \le p$, cid_i και $miles_i$, είναι το αναγνωριστικό και τα μίλια, αντίστοιχα, του i-οστού πελάτη, **σύμφωνα με διάσχιση κατά επίπεδα**.

X

Γεγονός τύπου **Print Airlines** το οποίο σηματοδοτεί την εκτύπωση όλων των αεροπορικών εταιρειών από τη λίστα αεροπορικών εταιρειών, καθώς και των αεροπλάνων που αποθηκεύονται στο δένδρο αεροπλάνων της κάθε εταιρείας.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
X
        Airline <aid_1> = <pid_1,1>, <pid_1,2>, …, <pid_1,m1 >
        Airline <aid_2> = <pid_2,1>, <pid_2,2>, …, <pid_2,m2 >
        ...
        Airline  <aid_n> = <pid_n,1 >, <pid_n,2>, ..., <pid_n,mn >
DONE
```

όπου n είναι το πλήθος των αεροπορικών εταιρειών στη λίστα αεροπορικών εταιρειών, για κάθε i, $1 \le i \le n$, aid_i είναι το αναγνωριστικό του i-οστού κόμβου στη λίστα των αεροπορικών εταιρειών, mi είναι το πλήθος των κόμβων της λίστας των αεροπλάνων της i-οστής εταιρείας στη λίστα και για κάθε $j \in \{1, ..., mi\}$, $pid_{i,j}$ είναι το αναγνωριστικό του αεροπλάνου που αντιστοιχεί στον j-οστό κόμβο του δένδρου των αεροπλάνων της i-οστής εταιρείας, σύμφωνα με την ενδοδιατεταγμένη διάσχιση.

Y

Γεγονός τύπου **Print Destinations** το οποίο σηματοδοτεί την εκτύπωση του πίνακα DESTINATIONS.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
Y
        Destination1 = <pid1,1 : t1,1>, ..., <pid1,n1 : t1,n1>
        …
        Destination10 = <pid10,1 : t10,1>, ..., <pid10,n10 : t10,n10>
DONE
```

όπου για κάθε j, $1 \le j \le 10$, n_j είναι το πλήθος των κόμβων στο δένδρο των πτήσεων του j-οστού προορισμού και για κάθε $i \in \{1, ..., n_j\}$, $pid_{j,i}$ και $t_{j,i}$ είναι το αναγνωριστικό του αεροπλάνου που εκτελεί την πτήση και η ώρα αναχώρησης (πεδίο depart_time) της πτήσης, αντίστοιχα, που αντιστοιχούν στον i-οστό κόμβο του δένδρου των πτήσεων του j-οστού προορισμού σύμφωνα με την ενδοδιατεταγμένη διάσχιση. Πρέπει να χρησιμοποιείστε τα νήματα για να κάνετε τη διάσχιση, ξεκινώνατας από τον αριστερότερο κόμβο του δένδρου και προχωρώντας πάντα στον επόμενο του στην ενδοδαιτεταγμένη διάσχιση.

Z

Γεγονός τύπου **Print Clients** το οποίο σηματοδοτεί την εκτύπωση όλων των πελατών από το δένδρο πελατών.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
Z
        Priority Queue = <cid1 : miles1>, <cid2 : miles2>, …, <cidp : milesp>
 DONE
```

όπου p είναι το πλήθος στοιχείων στην ουρά προτεραιότητας και για κάθε i, $1 \le i \le p$, cid_i και

$miles_i$, είναι το αναγνωριστικό και τα μίλια, αντίστοιχα, του i-οστού πελάτη, **σύμφωνα με διάσχιση κατά επίπεδα.** Για την υλοποίηση της διάσχισης κατά επίπεδα χρειάζεστε μια ουρά (FIFO queue). Εφόσον γνωρίζετε τα πλήθος των στοιχείων στην ουρά προτεραιότητας, η ουρά FIFO μπορεί να υλοποιηθεί με στατικό τρόπο (δηλαδή με πίνακα), όπως έχετε διδαχθεί στο μάθημα.

Δομές Δεδομένων (1/2)

Στην υλοποίησή σας δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες δομές δεδομένων (π.χ., ArrayList στη Java, κ.α.). Στη συνέχεια παρουσιάζονται οι δομές σε C που πρέπει να χρησιμοποιηθούν για την υλοποίηση της παρούσας εργασίας.

```
struct airline {
        int aid;
        struct airplane *pR;
        struct airlines *next;
        struct airlines *prev;
 }

struct airplane {
        int pid;
        int dest;   // [0,9]
        int depart_time;
        struct airplane *lc;
        struct airplane *rc;
}

struct flight {
        int pid;
        int depart_time;
        struct flight *lc;
        struct flight *rc;
        int thread_status;  // [0,3]
}

struct client {
        int cid;
        int miles;
        struct client *lc;
        struct client *mc;
        struct client *rc;
}

struct priority_queue{
        struct client *Q;
        int num_elements;
}
```

Δομές Δεδομένων (2/2)

```
struct fifo_queue{
        struct client *Q;               /* dynamically allocated array to store queue elements */
                                        /* used for level-order traversals of priority queue */
        int front;
        int length;
}

struct airline *airlines;               /* global variable, pointer to the head of the airline list */
struct client *clients;                 /* global variable, pointer to the root of a tree of clients */
struct flight * DESTINATIONS[10];       /* global variable, array of pointers to trees of flights*/
struct priority_queue *PQ;      /* global variable, pointer to a struct of type priority_queue */
struct fifo_queue *FQ;          /* global variable, pointer to a struct of type fifo_queue */
```

Βαθμολογία Γεγονότων

Code	Name	Points
R	**R**egister Airline	3
I	**I**nsert Airplane	5
C	**C**ancel Flight	8
D	**D**elete Airline	10
A	**A**cquisition Airline	12
S	**S**ubsidiary Company	15
N	**N**ew Client	10
E	**E**rase Client	10
T	**T**ravel Client	7
F	**F**ind Flight	7
X	Print Airlines	3
Y	Print Destinations	5
Z	Print Clients	5

ΚΑΛΗ ΕΠΙΤΥΧΙΑ