HY240: Δομές Δεδομένων
Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2017
Διδάσκουσα: Παναγιώτα Φατούρου
Προγραμματιστική Εργασία - 1ο Μέρος

Ημερομηνία Παράδοσης: Δευτέρα, 3 Απριλίου 2017, ώρα 23:59.

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 προσομοιώνει τη λειτουργία ενός συστήματος καταγραφής «σημείων ενδιαφέροντος» (π.χ. ξενοδοχεία, μουσεία, εστιατόρια, αξιοθέατα), από τουριστικής απόψεως, για ένα σύνολο τοποθεσιών. Οι χρήστες του συστήματος μπορούν να αντλήσουν από αυτό πληροφορίες για τις τοποθεσίες που σκοπεύουν να επισκεφτούν.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Το σύστημα αποτελείται από ένα σύνολο «σημείων ενδιαφέροντος» (points of interest (POI)) τα οποία είναι οργανωμένα ανά τοποθεσία. Για κάθε τοποθεσία, θεωρείστε δεδομένο ένα σημείο αναφοράς (π.χ. αυτό

μπορεί να είναι κάποιο κεντρικό σημείο της τοποθεσίας). Οι διαθέσιμες τοποθεσίες αποθηκεύονται σε μια απλά συνδεδεμένη λίστα, η οποία είναι **ταξινομημένη, σε αύξουσα διάταξη,** με βάση το αναγνωριστικό της κάθε τοποθεσίας. Η λίστα αυτή ονομάζεται **λίστα τοποθεσιών.** Ο κάθε κόμβος της λίστας είναι μια εγγραφή τύπου location (struct location) με τα ακόλουθα πεδία:

- lid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά την τοποθεσία.
- next: Δείκτης (τύπου location) στον επόμενο κόμβο της λίστας τοποθεσιών.
- poi_list: Ένας δείκτης (τύπου poi) στο πρώτο στοιχείο μιας διπλά συνδεδεμένης λίστας που ονομάζεται **λίστα σημείων ενδιαφέροντος της τοποθεσίας**. Η λίστα αυτή είναι **ταξινομημένη βάσει της απόστασης** του σημείου ενδιαφέροντος από το σημείο αναφοράς της κάθε τοποθεσίας. Κάθε στοιχείο της λίστας είναι μια εγγραφή τύπου poi (struct poi) με τα παρακάτω πεδία:
  - pid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το σημείο ενδιαφέροντος.
  - distance: Ένας αριθμός (τύπου int) που αντιστοιχεί στην απόσταση του σημείου ενδιαφέροντος με αναγνωριστικό pid **από το προηγούμενο σημείο ενδιαφέροντος της ίδιας τοποθεσίας**. Προσοχή για να υπολογίσετε την απόσταση ενός σημείου ενδιαφέροντος poi_x από το σημείο αναφοράς της τοποθεσίας όπου ανήκει πρέπει να αθροίσετε τις αποστάσεις όλων των σημείων ενδιαφέροντος που προηγούνται του poi_x στη λίστα.
  - type: Αναγνωριστικό (τύπου int) που αντιστοιχεί στην κατηγορία του σημείου ενδιαφέροντος. Η μεταβλητή αυτή λαμβάνει τιμές από 1 μέχρι 4, με κάθε τιμή να αντιστοιχεί σε μια κατηγορία (1: ξενοδοχείο, 2: μουσείο, 3: εστιατόριο, 4: αξιοθέατο)
  - prev: Δείκτης (τύπου poi) στον προηγούμενο κόμβο της λίστας σημείων ενδιαφέροντος της τοποθεσίας lid.
  - next: Δείκτης (τύπου poi) στον επόμενο κόμβο της λίστας σημείων ενδιαφέροντος της τοποθεσίας lid.

Στο Σχήμα 1 παρουσιάζεται η λίστα των τοποθεσιών, ο κάθε κόμβος της οποίας περιέχει μια λίστα σημείων ενδιαφέροντος.

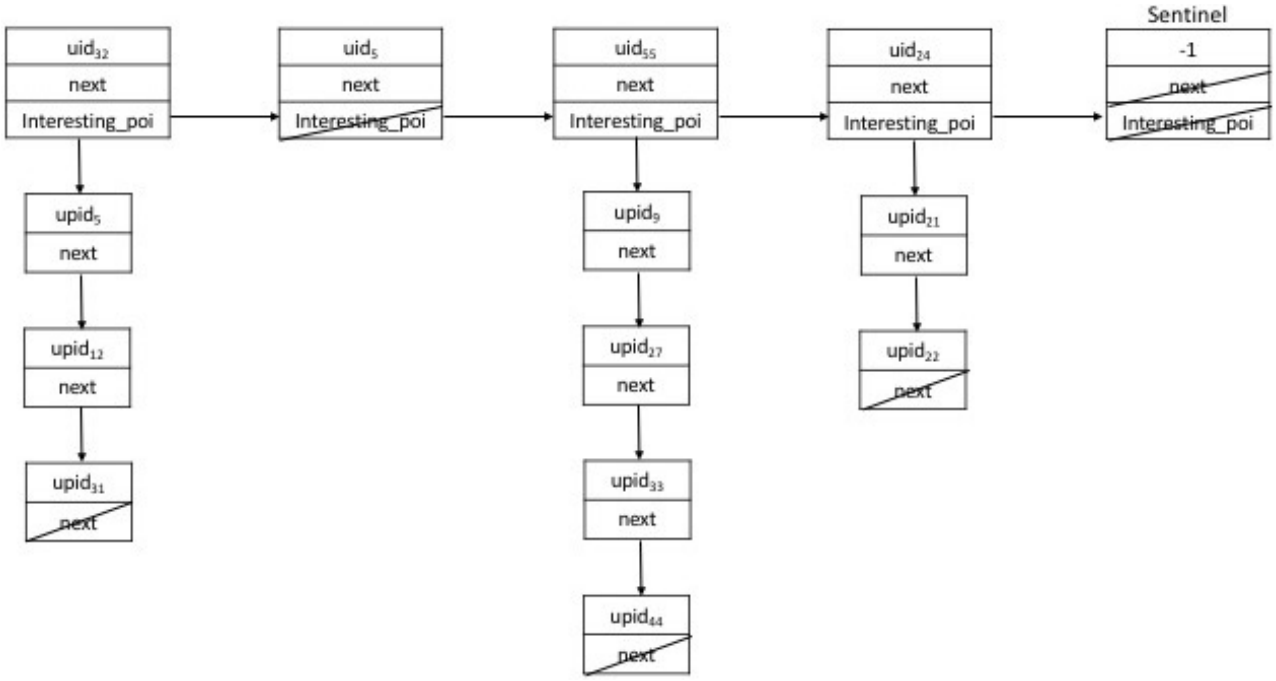
Σχήμα 1: Η απλά συνδεδεμένη λίστα τοποθεσιών, η οποία είναι ταξινομημένη σε αύξουσα διάταξη με βάση το αναγνωριστικό της κάθε τοποθεσίας και οι λίστες σημείων ενδιαφέροντος που δεικτοδοτούνται από κάθε κόμβο της. Η λίστα των σημείων ενδιαφέροντος κάθε τοποθεσίας είναι διπλά συνδεδεμένη και ταξινομημένη βάσει της απόστασης από το σημείο αναφοράς της τοποθεσίας.

Το σύστημα χρησιμοποιείται από ένα σύνολο χρηστών. Για το σκοπό αυτό θα δημιουργήσετε μια **απλά συνδεδεμένη, μη-ταξινομημένη λίστα με κόμβο φρουρό,** η οποία ονομάζεται **λίστα χρηστών**. Ο κάθε κόμβος της λίστας είναι μια εγγραφή τύπου user (struct user) με τα ακόλουθα πεδία:

- uid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον χρήστη.
- next: Δείκτης (τύπου poi) στον επόμενο κόμβο της λίστας χρηστών.
- interesting_poi: Δείκτης (τύπου user_poi *) στο πρώτο στοιχείο μιας απλά συνδεδεμένης λίστας, η οποία ονομάζεται **λίστα σημείων ενδιαφέροντος χρήστη**. Η λίστα αυτή περιέχει τα σημεία ενδιαφέροντος τα οποία σκοπεύει να επισκεφτεί ο χρήστης και είναι **ταξινομημένη σε αύξουσα διάταξη βάσει του αναγνωριστικού των σημείων ενδιαφέροντος.** Κάθε στοιχείο της λίστας είναι μια εγγραφή τύπου user_poi (struct user_poi) με τα παρακάτω πεδία:
  - upid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το σημείο ενδιαφέροντος.
  - next: Δείκτης (τύπου user_poi) στον επόμενο κόμβο της λίστας σημείων ενδιαφέροντος του χρήστη uid.

Ο κόμβος φρουρός της λίστας είναι ένας διαχειριστικός κόμβος για τον οποίο ισχύουν τα εξής: είναι και αυτός τύπου user με την ιδιαιτερότητα ότι το πεδίο uid έχει τιμή -1 και οι δείκτες interesting_poi και next

έχουν τιμή NULL (δηλαδή δεν χρησιμοποιούνται). Το Σχήμα 2 απεικονίζει τη λίστα χρηστών.



Σχήμα 2 Η μη ταξινομημένη λίστα χρηστών με κόμβο φρουρό. Ο κάθε κόμβος (χρήστης) περιέχει μια απλά συνδεδεμένη, ταξινομημένη λίστα σημείων ενδιαφέροντος χρήστη.

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 ακόλουθων μορφών:

– L <lid>

Γεγονός που υποδηλώνει την εισαγωγή μιας νέας τοποθεσίας με αναγνωριστικό <lid> στο σύστημα. Κατά το γεγονός αυτό θα γίνεται εισαγωγή ενός νέου κόμβου τύπου location στη λίστα τοποθεσιών. Μετά από κάθε εισαγωγή, η λίστα τοποθεσιών πρέπει να παραμένει ταξινομημένη. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
L <lid>
    Locations = <lid_1>, <lid_2>, ..., <lid_n>
DONE
```

όπου n είναι ο αριθμός των κόμβων στη λίστα τοποθεσιών και για κάθε i ∈ {1, …, n}, <lid_i> είναι το αναγνωριστικό της τοποθεσίας που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

– P <pid> <type> <distance> <lid>

Γεγονός που υποδηλώνει την εισαγωγή ενός νέου σημείου ενδιαφέροντος με αναγνωριστικό <pid> στην τοποθεσία με αναγνωριστικό <lid>. Η παράμετρος <type> αντιστοιχεί στον τύπο του σημείου ενδιαφέροντος όπως περιγράφεται παραπάνω στο πρώτο σκέλος της εκφώνησης. Η παράμετρος <distance> περιγράφει την απόσταση του νέου σημείου ενδιαφέροντος από το σημείο αναφοράς της τοποθεσίας <lid>. Το γεγονός αυτό προσθέτει το νέο σημείο ενδιαφέροντος στην κατάλληλη θέση βάσει του <distance> στη λίστα σημείων ενδιαφέροντος της τοποθεσίας <lid>. Είναι αξιοσημείωτο ότι το πεδίο distance της εγγραφής του νέου σημείου ενδιαφέροντος πρέπει να περιέχει την απόσταση του από το σημείο ενδιαφέροντος στην προηγούμενη θέση της λίστας και όχι την απόστασή του από το σημείο αναφοράς της τοποθεσίας. Ωστόσο στην περίπτωση που το σημείο ενδιαφέροντος είναι αυτό που βρίσκεται πιο κοντά στο σημείο αναφοράς της τοποθεσίας (δηλ. ο πρώτος κόμβος της λίστας), τότε το πεδίο distance είναι η απόστασή του από το σημείο αναφοράς. Σε περίπτωση που το νέο σημείο ενδιαφέροντος τοποθετείται ανάμεσα σε υπάρχοντα σημεία ενδιαφέροντος, τότε πρέπει να ενημερώνεται κατάλληλα το πεδίο distance του κόμβου που θα είναι ο επόμενος του νέου κόμβου μετά την εισαγωγή. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
P <pid> <lid> <distance>
   Location = <pid_1:type_1:distance_1>, <pid_2:type_2:distance_2>, … <pid_n:type_n:distance_n>
DONE
```

όπου n είναι ο αριθμός των κόμβων στη λίστα σημείων ενδιαφέροντος της τοποθεσίας με αναγνωριστικό <lid> και για κάθε i ∈ {1, …, n}, <pid_i> και <$type_i$> είναι το αναγνωριστικό και ο τύπος αντίστοιχα του σημείου ενδιαφέροντος που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής και <$distance_i$> η απόσταση του i-οστού κόμβου της λίστας από τον προηγούμενο κόμβο.

– **A <pid> <lid>**

Γεγονός που υποδηλώνει ότι ένα σημείο ενδιαφέροντος με αναγνωριστικό pid που βρίσκεται στην τοποθεσία με αναγνωριστικό lid δεν θα είναι πλεόν διαθέσιμο στους επισκέπτες. Κατά το γεγονός αυτό αρχικά θα αναζητήσετε στη λίστα τοποθεσιών την τοποθεσία με αναγνωριστικό lid και στη συνέχεια θα διαγράψετε από τη λίστα σημείων ενδιαφέροντός της, τον κόμβο με αναγνωριστικό pid. Μετά από τη διαγραφή, η λίστα των σημείων ενδιαφέροντος της τοποθεσίας πρέπει να παραμένει ταξινομημένη. Επίσης θα πρέπει να ανανεωθεί το πεδίο τη απόστασης του επόμενου κόμβου απ' αυτόν που διαγράφεται προσθέτοντας την απόσταση του κόμβου που θα διαγραφεί. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
A <pid> <lid>
   Location = <pid_1:type_1:distance_1>, <pid_2:type_2:distance_2>, … <pid_n:type_n:distance_n>
DONE
```

όπου n είναι ο αριθμός των κόμβων στη λίστα σημείων ενδιαφέροντος της τοποθεσίας με αναγνωριστικό <lid> και για κάθε i ∈ {1, …, n}, $<pid_i>$ και $<type_i>$ είναι το αναγνωριστικό και ο τύπος αντίστοιχα του σημείου ενδιαφέροντος που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής και $<distance_i>$ η απόσταση του i-οστού κόμβου της λίστας από τον προηγούμενο κόμβο.

– **R <uid>**

Γεγονός που υποδηλώνει την εγγραφή ενός νέου χρήση με αναγνωριστικό <uid> στο σύστημα. Κατά το γεγονός αυτό θα γίνεται εισαγωγή ενός νέου κόμβου τύπου user στη λίστα χρηστών. Η κάθε εισαγωγή πρέπει να πραγματοποιείται σε πολυπλοκότητα O(1). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
R <uid>
   Users = <uid_1>, <uid_2>, ..., <uid_n>
DONE
```

όπου n είναι ο αριθμός των κόμβων στη λίστα χρηστών και για κάθε i ∈ {1, …, n}, $<uid_i>$ είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

– **I <uid> <upid>**

Γεγονός που υποδηλώνει ότι ο χρήστης με αναγνωριστικό uid ενδιαφέρεται για το σημείο ενδιαφέροντος με αναγνωριστικό pid. Κατά το γεγονός αυτό αρχικά θα πρέπει να γίνεται αναζήτηση του χρήστη με αναγνωριστικό uid στη λίστα χρηστών. Στη συνέχεια θα γίνεται εισαγωγή του σημείου ενδιαφέροντος με αναγνωριστικό pid στη λίστα σημείων ενδιαφέροντος του χρήστη στην κατάλληλη θέση (ώστε η λίστα να παραμένει ταξινομημένη με βάση το αναγνωριστικό των σημείων ενδιαφέροντος). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
I <uid> <pid>
   POI = <pid₁>, <pid₂>, ..., <pidₙ>
DONE
```

όπου n είναι ο αριθμός των κόμβων στη λίστα σημείων ενδιαφέροντος του χρήστη με αναγνωριστικό uid και για κάθε i ∈ {1, …, n}, <pid_i> είναι το αναγνωριστικό του σημείου ενδιαφέροντος που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

– **G <uid_1> <uid_2> <uid_3>**

Γεγονός που υποδηλώνει τη δημιουργία μιας ομάδας τριών χρηστών με αναγνωριστικά uid_1, uid_2, και uid_3 οι οποίοι θα περιηγηθούν μαζί στα κοινά σημεία που τους ενδιαφέρουν. Κατά το γεγονός αυτό, αρχικά θα εντοπίσετε τους τρεις χρήστες στη λίστα χρηστών. Η διαδικασία αυτή θα πρέπει να επιτυγχάνεται με πολυπλοκότητα O(n), όπου n είναι ο αριθμός των κόμβων της λίστας χρηστών (δηλαδή η αναζήτηση των 3 χρηστών θα πρέπει να γίνεται με μία μόνο διάσχιση της λίστας). Στη συνέχεια θα πρέπει να εντοπίσετε τα κοινά σημεία ενδιαφέροντος από τις λίστες των τριών χρηστών. Η διαδικασία αυτή (τομή των τριών λιστών) πρέπει να επιτυγχάνεται με πολυπλοκότητα $O(m_1+ m_2+ m_3)$, όπου m_1, m_2 και m_3 είναι οι αριθμοί των κόμβων της λίστας ενδιαφέροντος του πρώτου, του δεύτερου και του τρίτου χρήστη, αντίστοιχα (ο αλγόριθμος υπολογισμού της τομής που θα σχεδιάσετε θα πρέπει να διατρέχει την καθεμιά από τις τρεις αυτές λίστες το πολύ μια φορά). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
G <uid₁> <uid₂> <uid₃>
   User₁ = <pid₁,₁>, <pid₁,₂>, ..., <pid₁,ₙ₁>
   User₂ = <pid₂,₁>, <pid₂,₂>, ..., <pid₂,ₙ₂>
   User₃ = <pid₃,₁>, <pid₃,₂>, ..., <pid₃,ₙ₃>
DONE
```

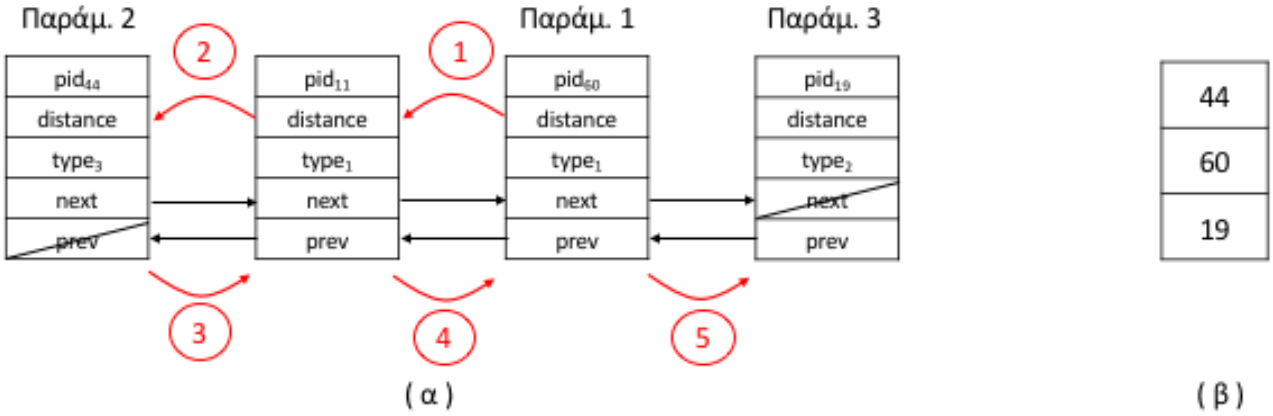
για κάθε j, $1 \leq j \leq 3$, n_j είναι το μέγεθος της λίστας σημείων ενδιαφέροντος του χρήστη που αντιστοιχεί στην j-οστή παράμετρο του γεγονότος, και για κάθε $i \in \{1 \ldots n_j\}$, $pid_{i,j}$ είναι το αναγνωριστικό του σημείου ενδιαφέροντος που αντιστοιχεί στον i-στο κόμβο της λίστας σημείων ενδιαφέροντος του j-οστού χρήστη.

– **B <lid> <pid_1> <pid_2> <pid_3>**

Γεγονός κατά το οποίο θα πρέπει να υπολογιστεί η απόσταση που πρέπει να διανύσει ένας επισκέπτης για να επισκεφτεί τα σημεία ενδιαφέροντος με αναγνωριστικά pid_1, pid_2 και pid_3 με τη σειρά που αυτά δίνονται. Κατά το γεγονός αυτό, αρχικά πρέπει να αναζητήσετε την τοποθεσία με αναγνωριστικό lid στη λίστα τοποθεσιών. Στη συνέχεια, θα εντοπίσετε τους κόμβους με αναγνωριστικά pid_1, pid_2 και pid_3 στη λίστα σημείων ενδιαφέροντος της τοποθεσίας, πραγματοποιώντας *ένα μόνο πέρασμα της λίστας αυτής*.

Γι' αυτό το γεγονός, θα πρέπει να γνωρίζετε την κατεύθυνση (προς τα αριστερά ή προς τα δεξιά) που πρέπει να ακολουθήσετε για να μεταβείτε από τον κόμβο με αναγνωριστικό pid_1 στον κόμβο με αναγνωριστικό pid_2 και στη συνέχεια στον κόμβο με αναγνωριστικό pid_3. Για το σκοπό αυτό θα χρησιμοποιήσετε έναν βοηθητικό πίνακα τριών θέσεων με περιεχόμενα **δείκτες τύπου poi**. Καθώς διατρέχετε τη λίστα σημείων ενδιαφέροντος

για να εντοπίσετε αυτά που δόθηκαν ως παράμετροι του γεγονότος, θα αποθηκεύετε στον βοηθητικό πίνακα δείκτες στους κόμβους που τους αντιστοιχούν με τη σειρά που συναντάτε αυτούς τους κόμβους στη λίστα. Για παράδειγμα, χρησιμοποιώντας τις εγγραφές του Σχήματος 1, θεωρήστε ότι οι παράμετροι του γεγονότος είναι: `B 20 60 44 19. Τότε,` αφού εντοπίσετε την τοποθεσία με αναγνωριστικό 20, θα επικεντρωθείτε στη λίστα με τα σημεία ενδιαφέροντος αυτής της τοποθεσίας όπως φαίνεται στο Σχήμα 3α. Καθώς διατρέχετε τη λίστα, κάθε φορά που συναντάτε κάποιο απ' τα σημεία με αναγνωριστικό 60, 44 ή 19 (τα οποία αποτελούν παραμέτρους του γεγονότος) θα αποθηκεύετε, στο βοηθητικό πίνακα, έναν δείκτη στον αντίστοιχο κόμβο με τη σειρά που συναντάτε τους κόμβους αυτούς. Στο τέλος αυτής της διαδικασίας, ο πίνακας του παραδείγματος θα περιέχει δείκτες στους κόμβους που αντιστοιχούν στα σημεία αυτά με την ακόλουθη σειρά: 44, 60, 19 (όπως φαίνεται στο Σχήμα 3β, όπου π.χ. η σημειογραφία $\vec{44}$ υποδηλώνει έναν δείκτη στον κόμβο που αντιστοιχεί στο σημείο ενδιαφέροντος με αναγνωριστικό 44).



Σχήμα 3

Στη συνέχεια χρησιμοποιώντας το δείκτη στο σημείο ενδιαφέροντος της πρώτης παραμέτρου θα εξετάσετε αν πρέπει να διασχίσετε τη λίστα προς τα εμπρός ή προς τα πίσω. Για το σκοπό αυτό θα ελέγχετε αν το αναγνωριστικό του πρώτου σημείου ενδιαφέροντος βρίσκεται σε προγενέστερη ή μεταγενέστερη θέση του πίνακα σε σχέση με το αναγνωριστικό της δεύτερης παραμέτρου. Αν είναι σε μεταγενέστερη θέση, τότε σημαίνει ότι πρέπει να διασχίσετε τη λίστα προς τα αριστερά (ακολουθώντας δείκτες prev) ενώ αν είναι σε προγενέστερη θέση, θα πρέπει να προχωρήσετε στη λίστα προς τα εμπρός. Στη συνέχεια, επαναλαμβάνετε την ίδια διαδικασία για το δεύτερο και τρίτο σημείο ενδιαφέροντος που έχουν δοθεί ως παράμετροι στο γεγονός.

Στο παραπάνω παράδειγμα το πρώτο σημείο ενδιαφέροντος που δίνεται ως παράμετρος είναι το 60, το οποίο βρίσκεται στη δεύτερη θέση του βοηθητικού πίνακα. Το δεύτερο σημείο ενδιαφέροντος είναι το 44 που βρίσκεται στην πρώτη θέση του βοηθητικού πίνακα. Αυτό σημαίνει ότι ξεκινώντας από το σημείο ενδιαφέροντος με αναγνωριστικό 60, πρέπει να ακολουθήσουμε δείκτες prev για να εντοπίσουμε τον κόμβο με αναγνωριστικό 44 (όπως φαίνονται στα δύο πρώτα βήματα της εικόνας 3α). Στη συνέχεια εξετάζουμε το τρίτο σημείο ενδιαφέροντος με αναγνωριστικό 19, το οποίο βρίσκεται στην τρίτη θέση του βοηθητικού πίνακα. Αυτό σημαίνει ότι το δεύτερο σημείο βρίσκεται σε προγενέστερη θέση σε σχέση με το τρίτο σημείο. Αυτό σημαίνει ότι πρέπει να προχωρήσουμε στη λίστα ακολουθώντας δείκτες next από το σημείο 44 για να συναντήσουμε το σημείο 19 (βήματα 3-5 στην εικόνα 3α).

Για να υπολογίσετε την απόσταση που πρέπει να διανύσει ένας επισκέπτης για να μεταβεί από το ένα σημείο ενδιαφέροντος στο άλλο όπως ορίζονται από τις παραμέτρους του γεγονότος, πρέπει να μετράτε και τις

αποστάσεις των ενδιάμεσων κόμβων.

Στο παραπάνω παράδειγμα για να μεταβεί κάποιος από το σημείο ενδιαφέροντος 60 στο 44 πρέπει να περάσει από αυτό με αναγνωριστικό 11. Άρα πρέπει να αθροίσετε την απόσταση από το 60 στο 11 και την απόσταση από το 11 στο 44. Αντίστοιχα για να μεταβεί ο επισκέπτης από το σημείο 44 στο 19 πρέπει να αθροίσετε την απόσταση από το 44 στο 11, την απόσταση από το 11 στο 60 και από το 60 στο 19.

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

```
B <lid> <pid1> <pid2> <pid3>
   Total distance: <distance>
DONE
```

– **X**

Γεγονός τύπου print locations το οποίο σηματοδοτεί την εκτύπωση όλων των τοποθεσιών από τη λίστα τοποθεσιών. Για την κάθε τοποθεσία θα πρέπει να εκτυπώνονται όλα τα σημεία ενδιαφέροντός του. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
X
 LOCATIONS:
   Location1 = <pid1,1:type1,1:distance1,1>, . . . <pid1,n1:type1,n1:distance1,n1>
   Location2 = <pid2,1:type2,1:distance2,1>, . . . <pid2,n2:type2,n2:distance2,n2>
   . . .
   Locationk = <pidk,1:typek,1:distancek,1>, . . . <pidk,nk:typek,nk:distancek,nk>
DONE
```

όπου k είναι ο αριθμός των κόμβων στη λίστα τοποθεσιών, για κάθε j, $1 \leq j \leq k$, n_j είναι το μέγεθος της λίστας σημείων ενδιαφέροντος της j-οστής τοποθεσίας, και για κάθε $\iota \in \{1 \ldots n_j\}$, $pid_{j,i}$, $type_{j,i}$ και $distance_{j,i}$ είναι το αναγνωριστικό, το είδος και η απόσταση αντίστοιχα του σημείου ενδιαφέροντος που αντιστοιχεί στον i-στο κόμβο της λίστας της j-οστής τοποθεσίας.

– **Y**

Γεγονός τύπου print users το οποίο σηματοδοτεί εκτύπωση όλων των χρηστών της λίστας χρηστών. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
Y
   User1: <pid1,1>, <pid1,2>, . . . <pid1,n1>
   User2: <pid2,1>, <pid2,2>, . . . <pid2,n2>
   . . .
   Userk: <pidk,1>, <pidk,2>, . . . <pidk,nk>
DONE
```

όπου k είναι ο αριθμός των κόμβων στη λίστα χρηστών, για κάθε j, $1 \leq j \leq k$, n_j είναι το μέγεθος της λίστας

σημείων ενδιαφέροντος του j-οστού χρήστη, και για κάθε ι ∈ {1 … n_j}, $pid_{j,i}$, είναι το αναγνωριστικό του σημείου ενδιαφέροντος που αντιστοιχεί στον i-στο κόμβο της λίστας του j-οστού χρήστη.

- **Z <uid>**

Γεγονός τύπου search user το οποίο σηματοδοτεί την αναζήτηση του χρήστη με αναγνωριστικό uid στη λίστα χρηστών. Όταν ο χρήστης εντοπιστεί θα πρέπει να εκτυπώνετε το ξενοδοχείο που διαμένει αλλά και τη λίστα με τα σημεία ενδιαφέροντός του. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
Z <uid>
    Hotel: <hotel-id>
    Interesting_poi: <poi1>, <poi2>, . . . <poin>,
DONE
```

όπου n είναι το μέγεθος της λίστας σημείων ενδιαφέροντος του χρήστη με αναγνωριστικό uid και για κάθε i ∈ {1, …, n}, <pid_i> είναι το αναγνωριστικό του σημείου ενδιαφέροντος που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

Βαθμολογία γεγονότων

L	10
P	15
A	10
R	10
I	12
G	15
B	16
X	4
Y	4
Z	4

Δομές Δεδομένων

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

```
struct location {
	int lid;
	struct * poi poi_list;
	struct location * next;
};

struct poi {
	int pid;
	int distance;
	int type
	struct poi * next;
	struct poi * prev;
};

struct user {
	int uid;
	struct user_poi * interesting_poi;
	struct user * next;
};

struct user_poi {
	int upid;
	struct user_poi * next;
};

/* global variable, pointer to the beginning of the locations list*/
struct location * locations_list;

/* global variable, pointer to the beginning of the users list*/
struct user * users_list;

/* global variable, pointer to the sentinel node of the users list */
struct user * users_sentinel;
```