HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2016-2017
Διδάσκουσα: Παναγιώτα Φατούρου
Προγραμματιστική Εργασία - 2ο Μέρος

Ημερομηνία Παράδοσης: Δευτέρα, 19 Δεκεμβρίου 2016, ώρα 23:59.

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

Γενική Περιγραφή

Στην εργασία αυτή καλείστε να υλοποιήσετε ένα πρόγραμμα που προσομοιώνει τη λειτουργία ενός ζωολογικού πάρκου. Το πάρκο αυτό αποτελείται από ένα σύνολο οικοσυστημάτων τα οποία φιλοξενούν διάφορα είδη ζώων και είναι ανοιχτό προς τους επισκέπτες. Στο πάρκο απασχολούνται κάποιοι εργαζόμενοι στους οποίους έχει ανατεθεί η επιμέλεια και φροντίδα κάποιων ζώων που φιλοξενούνται στα διάφορα οικοσυστήματα.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Στο ζωολογικό πάρκο κατοικούν διάφορα είδη ζώων τα οποία συνυπάρχουν ομαδοποιημένα σε οικοσυστήματα τα οποία τους επιτρέπουν να συμβιώνουν αρμονικά.

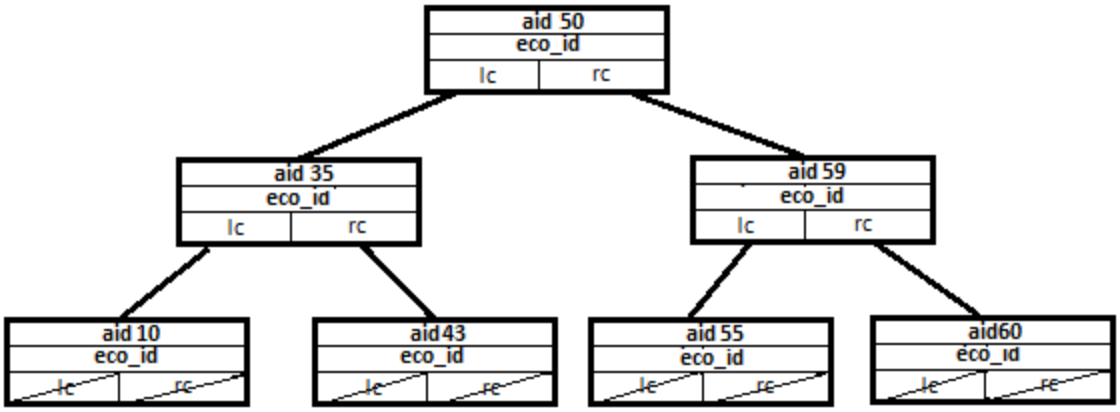
Όλα τα ζώα του πάρκου αποθηκεύονται σε ένα **απλά συνδεδεμένο δένδρο δυαδικής αναζήτησης** (Binary Search Tree), το οποίο είναι **ταξινομημένο σε αύξουσα διάταξη** με βάση το αναγνωριστικό του κάθε ζώου. Το δένδρο αυτό ονομάζεται **δένδρο ζώων** και ο κάθε κόμβος του αποτελεί μια εγγραφή τύπου animal με τα ακόλουθα πεδία:

- **aid:** Αναγνωριστικό (τύπου **int**) που χαρακτηρίζει μοναδικά το ζώο.
- **lc:** Δείκτης (τύπου **animal**) στο αριστερό παιδί του κόμβου στο δένδρο ζώων.
- **rc**: Δείκτης (τύπου **animal**) στο δεξιό παιδί του κόμβου στο δένδρο ζώων.

Στην δεύτερη φάση της προγραμματιστικής εργασίας, θα κάνουμε την θεώρηση ότι το κάθε οικοσύστημα μπορεί να φιλοξενήσει το πολύ 20 ζώα. Συνεπώς, μπορούμε να δεσμεύσουμε ένα εύρος 20 αναγνωριστικών για

τα ζώα κάθε οικοσυστήματος. Για παράδειγμα, τα ζώα του πρώτου οικοσυστήματος θα έχουν αναγνωριστικά στο διάστημα 0-19, του δεύτερου οικοσυστήματος στο διάστημα 20-39, κ.ο.κ.

Στο Σχήμα 1 παρουσιάζεται το δένδρο όλων των ζώων του ζωολογικού πάρκου.

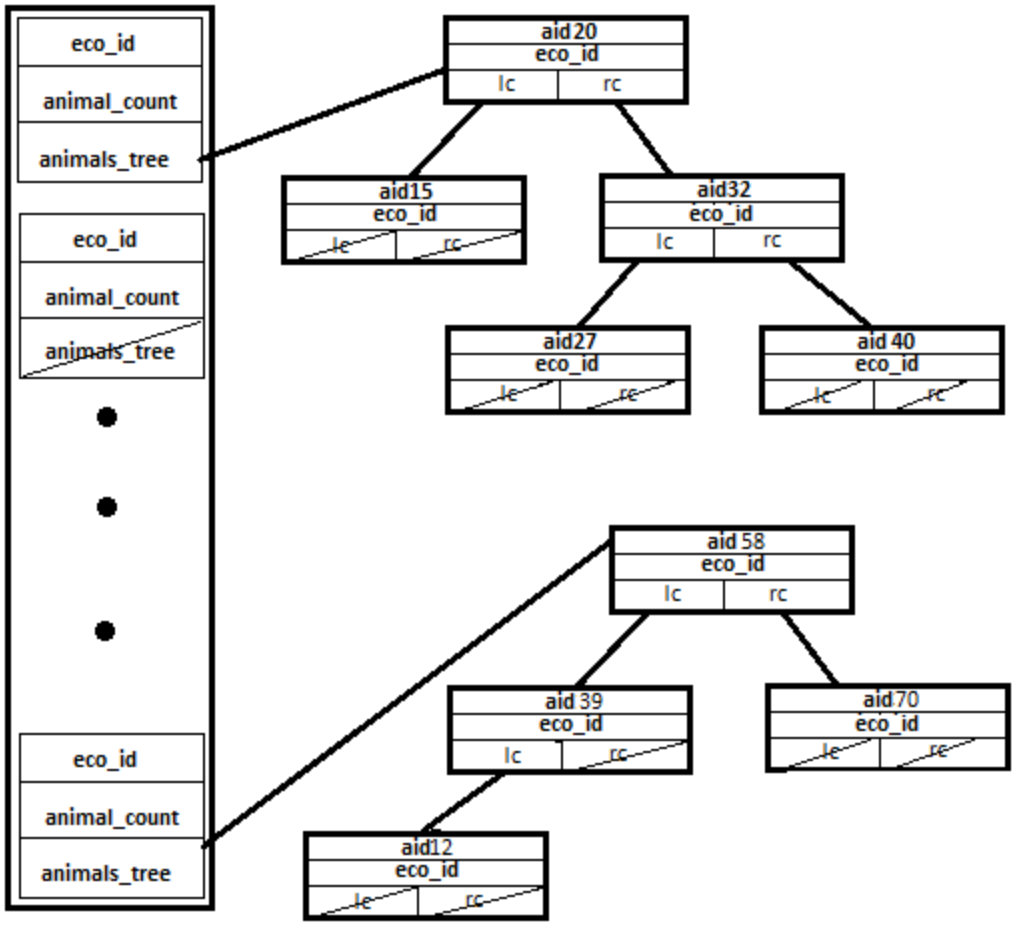


Σχήμα 1: Το δυαδικό δένδρο αναζήτησης ζώων όλου του πάρκου

Για τη διαχείριση των οικοσυστημάτων, θα δημιουργήσετε έναν πίνακα σταθερού μεγέθους **50 θέσεων** (όσα και τα οικοσυστήματα του πάρκου), ο οποίος θα ονομάζεται **πίνακας οικοσυστημάτων**. Κάθε θέση του πίνακα αποτελεί μια εγγραφή τύπου ecosystem με τα ακόλουθα πεδία:

- **eco_id:** Αναγνωριστικό (τύπου **int**) που χαρακτηρίζει μοναδικά το οικοσύστημα
- **animals_tree:** Δείκτης (τύπου **animal**) στην ρίζα ενός **απλά συνδεδεμένου δένδρου δυαδικής αναζήτησης**, το οποίο ονομάζεται **δένδρο ζώων του οικοσυστήματος**. Το δένδρο αυτό περιέχει κόμβους τύπου **animal** και είναι **ταξινομημένο με βάση το αναγνωριστικό των ζώων**.

Στο Σχήμα 2 παρουσιάζεται ο πίνακας οικοσυστημάτων, και το δένδρο των ζώων που δεικτοδοτείται από κάθε στοιχείο του.



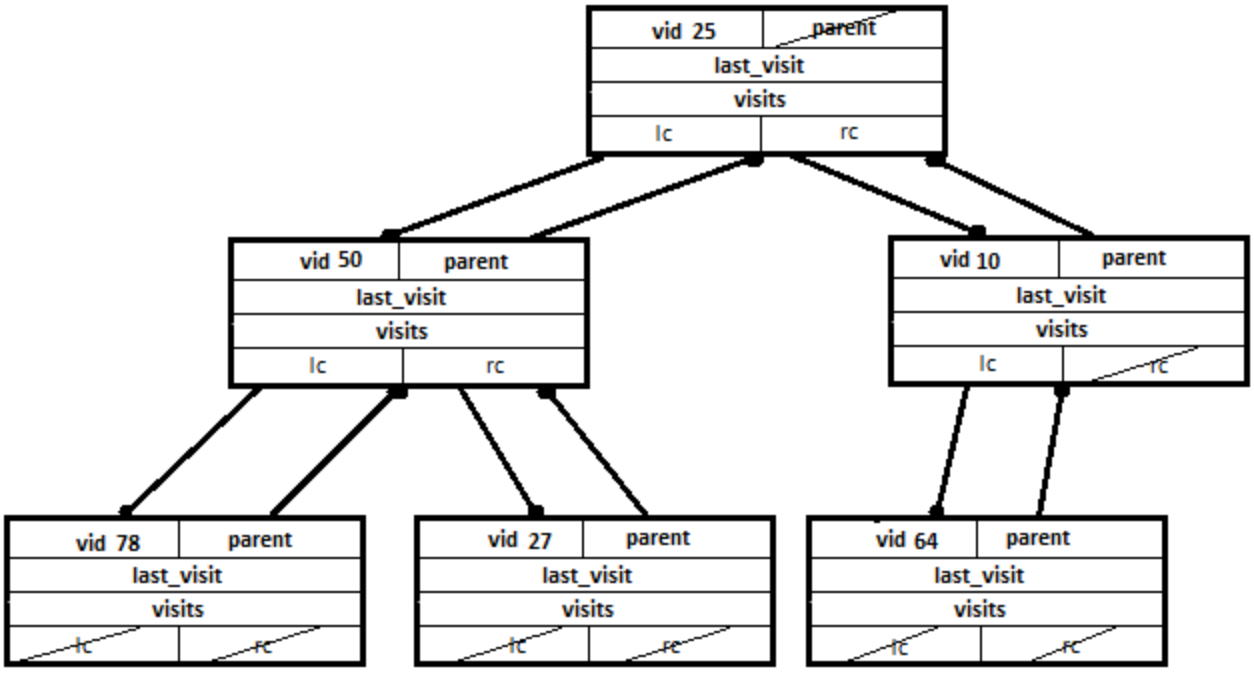
Σχήμα 2: Ο πίνακας οικοσυστημάτων με τα επιμέρους δυαδικά δένδρα των ζώων

Το πάρκο είναι ανοιχτό προς τους πολίτες και δεν είναι λίγοι αυτοί που το επισκέπτονται καθημερινά. Για το σκοπό αυτό θα δημιουργήσετε ένα **πλήρες, μη-ταξινομημένο, διπλά συνδεδεμένο, δυαδικό δένδρο,** το οποίο ονομάζεται **δένδρο επισκεπτών**. Ο κάθε κόμβος του δένδρου περιέχει μια εγγραφή τύπου **visitor** με τα ακόλουθα πεδία:

- **vid:** Αναγνωριστικό (τύπου **int**) που χαρακτηρίζει μοναδικά τον επισκέπτη του πάρκου.
- **last_visit:** Πεδίο (τύπου **int**) που αντιστοιχεί στο έτος της τελευταίας επίσκεψής του στο πάρκο.
- **visits:** Μετρητής (τύπου **int**) που αντικατοπτρίζει τον αριθμό των επισκέψεων του επισκέπτη στο πάρκο.
- **p:** Δείκτης (τύπου **visitor**) στον πατρικό κόμβο στο δένδρο επισκεπτών.
- **lc:** Δείκτης (τύπου **visitor**) στο αριστερό παιδί του κόμβου στο δένδρο επισκεπτών.
- **rc:** Δείκτης (τύπου **visitor**) στο δεξιό παιδί του κόμβου στο δένδρο επισκεπτών.

Για την δομή αυτή θα διατηρείτε δύο δείκτες. Έναν προς την ρίζα του δένδρου κι έναν προς το δεξιότερο φύλλο του δένδρου (rightmost leaf).

Το Σχήμα 3 απεικονίζει το δένδρο επισκεπτών.



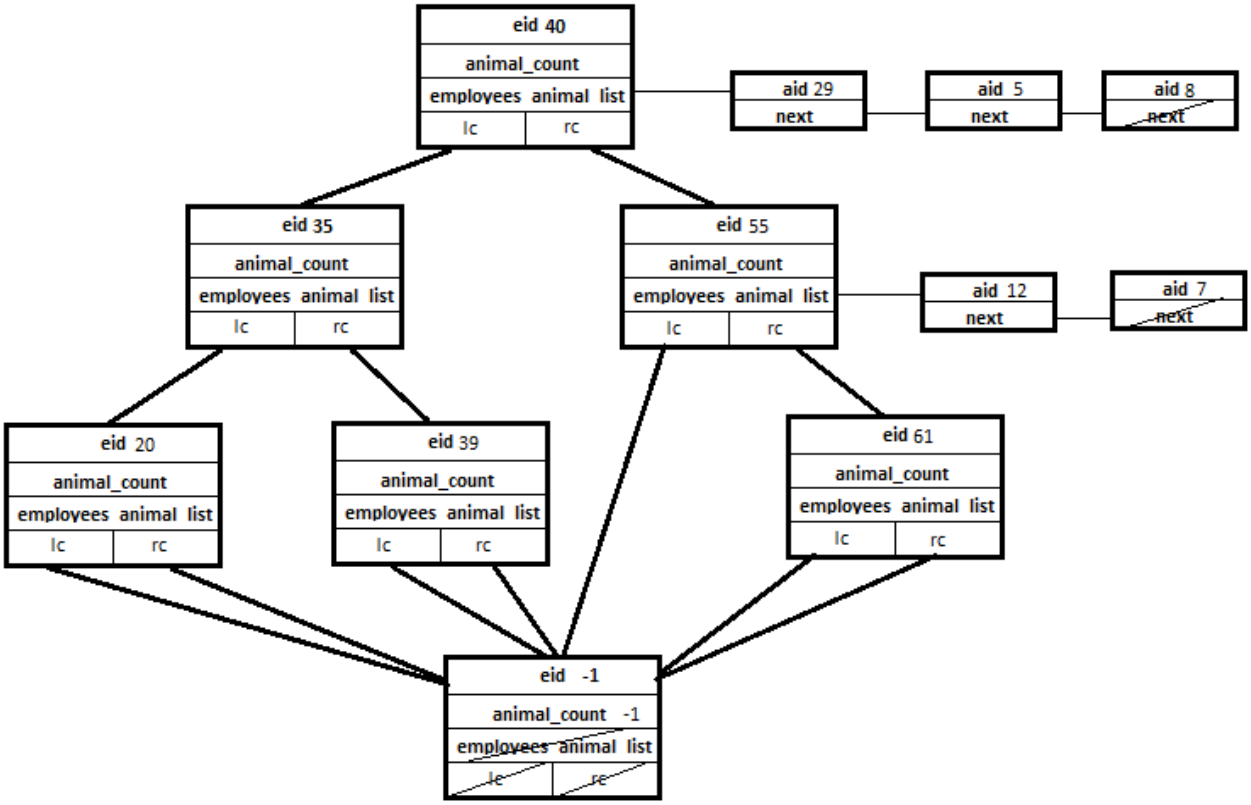
Σχήμα 3: Το δένδρο επισκεπτών

Οι εργαζόμενοι στο ζωολογικό πάρκο είναι υπεύθυνοι για την επιμέλεια **συγκεκριμένων ζώων** (όχι οικοσυστημάτων όπως στην πρώτη φάση). Για τον σκοπό αυτό θα δημιουργήσετε ένα **απλά συνδεδεμένο δυαδικό δένδρο, ταξινομημένο ως προς το αναγνωριστικό των εργαζομένων με κόμβο φρουρό** που ονομάζεται **δένδρο εργαζομένων**. Οι εγγραφές του δένδρου αυτού είναι τύπου **employee** και περιέχουν τα ακόλουθα πεδία:

- **eid:** Αναγνωριστικό (τύπου **int**) που χαρακτηρίζει μοναδικά τον κάθε εργαζόμενο του πάρκου.
- **animal_count:** Μετρητής (τύπου **int**) που περιγράφει τον αριθμό των ζώων για τα οποία είναι υπεύθυνος ο εργαζόμενος.
- **lc**: Δείκτης (τύπου **employee**) στο αριστερό παιδί του εργαζομένου στο δένδρο εργαζομένων.
- **rc**: Δείκτης (τύπου **employee**) στον δεξιό παιδί του εργαζομένου στο δένδρο εργαζομένων.
- **employees_animal_list:** Δείκτης (τύπου **employee_animal**) στο πρώτο στοιχείο μιας **απλά συνδεδεμένης λίστας** η οποία περιέχει τα ζώα για τα οποία είναι υπεύθυνος ο εργαζόμενος. Πιο συγκεκριμένα η δομή **employee_animal** περιέχει τα ακόλουθα πεδία:
  - **aid:** Το αναγνωριστικό του ζώου για το οποίο είναι υπεύθυνος ο εργαζόμενος
  - **next**: Δείκτης (τύπου **employee_animal**) στο επόμενο στοιχείο της λίστας

Ο κόμβος φρουρός του δένδρου είναι ένας διαχειριστικός κόμβος για τον οποίο ισχύουν τα εξής: είναι και αυτός τύπου **employee** με την ιδιαιτερότητα ότι το αναγνωριστικό του (**eid**) και το πεδίο **animal_count** έχει τιμή -1. Επίσης, τα πεδία lc και rc του κόμβου φρουρού είναι ίσα με NULL.

Στο Σχήμα 4 φαίνεται το δένδρο των εργαζομένων.



Σχήμα 4: Το δένδρο των εργαζομένων με τις λίστες ζώων του καθενός

Προκειμένου να έχουμε και την αντίστροφη αντιστοίχιση και να μπορούμε να βρούμε ποιος εργαζόμενος είναι υπεύθυνος για ένα συγκεκριμένο ζώο θα πρέπει να υλοποιηθεί ένας πίνακας κατακερματισμού **ANIMALS_TO_EMPLOYEES[hash_table_size]**, το μέγεθος του οποίου θα πρέπει να επιλέξετε εσείς προσεκτικά και να είστε σε θέση να δικαιολογήσετε την επιλογή σας.

Για την επίλυση των συγκρούσεων θα ακολουθήσετε την μέθοδο των **ταξινομημένων αλυσίδων.** Κάθε στοιχείο του πίνακα κατακερματισμού περιέχει έναν δείκτη στο πρώτο στοιχείο μιας **απλά συνδεδεμένης αλυσίδας** η οποία θα είναι **ταξινομημένη** ως προς το αναγνωριστικό του ζώου. Κάθε στοιχείο μιας αλυσίδας είναι μια εγγραφή τύπου **animal_to_employee** με τα εξής πεδία:

- **aid:** Το αναγνωριστικό του ζώου που αναζητούμε
- **eid:** Το αναγνωριστικό (τύποτ **int**) του εργαζομένου που είναι υπεύθυνος για το συγκεκριμένο ζώο
- **next**: Δείκτης (τύπου animal_to_employee) στο επόμενο στοιχείο της αλυσίδας

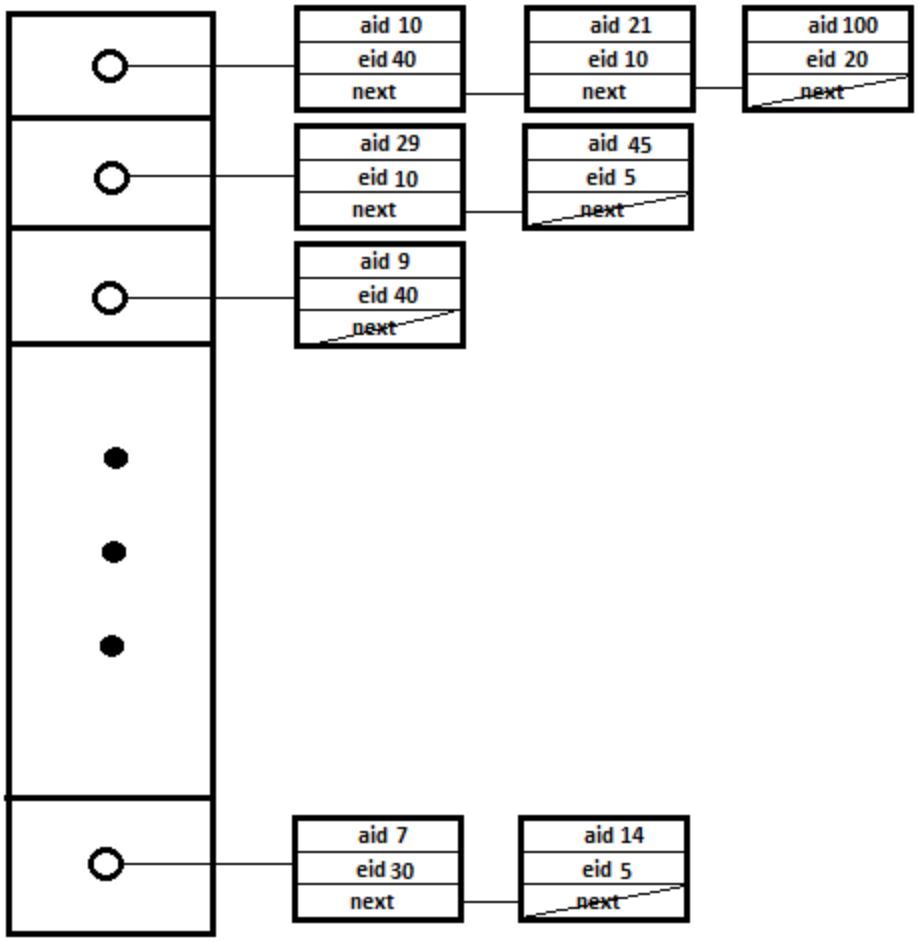
Πιο συγκεκριμένα ο κατακερματισμός θα γίνεται με βάση το αναγνωριστικό του εκάστοτε ζώου (**aid**) και στην συνέχεια θα αναζητείται η κατάλληλη εγγραφή στην αλυσίδα της αντίστοιχης θέσης του πίνακα κατακερματισμού. Σημειώνεται ότι το aid κάθε κόμβου της αλυσίδας που δεικτοδοτείται από τη θέση i του πίνακα κατακερματισμού, έχει τιμή κατακερματισμού i.

Για την υλοποίηση της συνάρτησης κατακερματισμού θα πρέπει να βασιστείτε στην τεχνική του καθολικού κατακερματισμού. Για την υλοποίηση του καθολικού κατακερματισμού θα δίνονται τα εξής:

1. Ένας πίνακας **primes**[], ο οποίος περιέχει πρώτους αριθμούς σε αύξουσα σειρά.
2. Το μέγιστο αναγνωριστικό ζώου **aid**, μέσω της μεταβλητής **max_animal_id**.
3. Το μέγιστο πλήθος ζώων, μέσω της μεταβλητής **animals_total_count.**

Αυτές οι μεταβλητές είναι global κι έχουν δηλωθεί στο αρχείο .h.

Στο σχήμα 5 φαίνεται ο πίνακας κατακερματισμού και οι επιμέρους αλυσίδες του.



Σχήμα 5: Ο πίνακας κατακερματισμού

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 <aid>

Γεγονός που υποδηλώνει ότι το ζώο με αναγνωριστικό **aid** θα φιλοξενηθεί στο ζωολογικό πάρκο. Κατά το γεγονός αυτό θα γίνεται εισαγωγή ενός νέου κόμβου τύπου **animal** στο δένδρο των ζώων σε χρόνο O(h), όπου h

είναι το ύψος του δένδρου. Μετά από κάθε εισαγωγή, το δένδρο των ζώων πρέπει να παραμένει ταξινομημένο. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
L <aid>
    Animals = <aid_1>, <aid_2>, ..., <aid_n>
DONE
```

όπου n είναι ο αριθμός των κόμβων του δένδρου των ζώων.

- D

Γεγονός τύπου distribute animals το οποίο σηματοδοτεί τον διαχωρισμό των ζώων ανά οικοσύστημα και τη μεταφορά των ζώων κάθε οικοσυστήματος στο δένδρο ζώων του οικοσυστήματος το οποίο δεικτοδοτείται από το κατάλληλο στοιχείο του πίνακα οικοσυστημάτων. Ο αλγόριθμος για τον διαχωρισμό των ζώων του δένδρου ζώων, θα πρέπει να υλοποιηθεί ως εξής:

Για κάθε σύνολο ζώων που ανήκουν σε ένα οικοσύστημα θα πρέπει να αναζητήσετε το ανώτατο αναγνωριστικό ζώου που αντιστοιχεί στο οικοσύστημα αυτό (π.χ. για το οικοσύστημα 1 θα πρέπει να αναζητήσετε το αναγνωριστικό ζώου 19, για το οικοσύστημα 2 το αναγνωριστικό 39 κλπ.) και να αποκόβετε τους δείκτες που βρίσκονται δεξιά κι αριστερά του μονοπατιού. Στην συνέχεια, τα δένδρα του δάσους που προκύπτει από τις παραπάνω αποκοπές δεικτών, θα πρέπει να συγχωνευθούν ώστε να προκύψουν δύο δένδρα, το T_i, το οποίο θα περιέχει τα ζώα με αναγνωριστικά μικρότερα ή ίσα του αναγνωριστικού που αναζητήθηκε και το οποίο θα τοποθετηθεί στην αντίστοιχη εγγραφή του πίνακα οικοσυστημάτων στην θέση *i* και το δένδρο T' το οποίο θα περιέχει τα ζώα με αναγνωριστικά μεγαλύτερα του αναγνωριστικού που αναζητήθηκε. Το T' θα χρησιμοποιείται κάθε φορά για να παραχθεί το δένδρο του επόμενου οικοσυστήματος. Η χρονική πολυπλοκότητα του αλγορίθμου θα πρέπει να είναι **O(h*<αριθμός_οικοσυστημάτων>),** όπου h είναι το ύψος του δένδρου ζώων αρχικά και ο αριθμός οικοσυστημάτων είναι 50.

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

```
D
    ECOSYSTEMS:
    <ecosystem_1>: <aid_1,1> . . . <aid_1,n1>
    <ecosystem_2>: <aid_2,1> . . . <aid_2,n2>
    ...
   <ecosystem_50>: <aid_50,1> . . . <aid_50,n50>
DONE
```

όπου για κάθε i, $1 \leq i \leq 50$, n_i είναι το μέγεθος του δένδρου ζώων του i-οστού οικοσυστήματος του πίνακα οικοσυστημάτων, και για κάθε j, $1 \leq j \leq n_i$, $<aid_{i,j}>$ είναι το αναγνωριστικό του j-οστού κόμβου στο δένδρο ζώων του i-οστού οικοσυστήματος, όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του. Οι κόμβοι θα πρέπει να έχουν εισαχθεί με τέτοιο τρόπο ώστε αν πραγματοποιηθεί ενδοδιατεταγμένη (in-order) διάσχιση στο δένδρο, τα ζώα να προσπελάζονται σε αύξουσα διάταξη ως προς το πεδίο aid.

- V <vid> <year>

Γεγονός τύπου visit το οποίο σηματοδοτεί την επίσκεψη ενός επισκέπτη με αναγνωριστικό **vid** στο ζωολογικό πάρκο το έτος **year**. Κατά το γεγονός αυτό αρχικά θα πρέπει να γίνεται αναζήτηση του επισκέπτη με αναγνωριστικό **vid** στο δένδρο επισκεπτών. Αν αυτός βρεθεί τότε πρέπει να γίνει ανανέωση του πεδίου **last_visit** με την τιμή **<year>** και να αυξηθεί ο μετρητής **visit_count**. Αν δεν βρεθεί κόμβος με αναγνωριστικό **vid** στο δένδρο, τότε πρέπει να εισάγετε έναν νέο κόμβο με αυτό το αναγνωριστικό στο δένδρο επισκεπτών. Τα πεδία **last_visit** και **visit_count** του νέου κόμβου θα πρέπει να έχουν τιμές <**year**> και 1, αντίστοιχα.

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

```
V <vid> <year>
Level0:       <vid0,1 : visits0,1 : last_visit0,1>
Level1:       <vid1,1 : visits1,1 : last_visit1,1> … <vid1,n1 : visits1,n1 : last_visit1,n1>
.
.
Levelh:       <vidh,1 : visitsh,1 : last_visith,1> … <vidh,nh : visitsh,nh : last_visith,nh>
DONE
```

Όπου για κάθε i, $0 \le i < h$, h το ύψος του δένδρου, n_i ο αριθμός των κόμβων στο επίπεδο i και για κάθε j, $1 \le j \le n_i$, και $<vid_{i,j} : visits_{i,j} : last\_visit_{i,j}>$ τα πεδία του του j-οστού κόμβου στο επίπεδο i του δένδρου. Πιο συγκεκριμένα, για την εκτύπωση του δένδρου επισκεπτών θα πρέπει να το διατρέξετε κατά πλάτος (Breadth-first) και να το τυπώσετε ανά επίπεδα. Για να το πετύχετε αυτό θα χρησιμοποιήσετε μία ουρά (queue) στην οποία θα αποθηκεύετε τους κόμβους του κάθε επιπέδου από αριστερά προς τα δεξιά. Είναι αξιοσημείωτο ότι θα πρέπει να εισάγετε dummy κόμβους στην ουρά για την δεικτοδότηση των επιπέδων του δένδρου, ώστε να τυπώνετε το σωστό επίπεδο σε κάθε γραμμή, π.χ. $Level_0$, $Level_1$ κλπ.

- O <years_interval>

Γεγονός τύπου delete old visitors το οποίο σηματοδοτεί τη διαγραφή των επισκεπτών από το δένδρο επισκεπτών για τους οποίους ισχύει ότι η τελευταία επίσκεψή τους στο πάρκο είχε πραγματοποιηθεί πριν από περισσότερα των **<years_interval>** έτη. Για να υπολογίσετε πόσα χρόνια έχουν περάσει από την τελευταία επίσκεψη αρκεί να γίνει η αφαίρεση από το τρέχον έτος (2016) του έτους της τελευταίας επίσκεψης του επισκέπτη (πεδίο **last_visit**).

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

```
O <years_interval>
Level0:       <vid0,1 : visits0,1 : last_visit0,1>
Level1:       <vid1,1 : visits1,1 : last_visit1,1> . . . <vid1,n1 : visits1,n1 : last_visit1,n1>
.
.
Levelh:       <vidh,1 : visitsh,1 : last_visith,1> … <vidh,nh : visitsh,nh : last_visith,nh>
DONE
```

όπου για κάθε i, $0 < i < h$, h το ύψος του δένδρου, n_i ο αριθμός των κόμβων στο επίπεδο i και για κάθε j, $1 \le j \le n_i$, και $<vid_{i,j} : visits_{i,j} : last\_visit_{i,j}>$ τα πεδία του του j-οστού κόμβου στο επίπεδο i του δένδρου. Η εκτύπωση του δένδρου θα πρέπει να γίνει κι εδώ ανά επίπεδα με την μέθοδο που περιγράφηκε παραπάνω.

- H <eid>

Γεγονός τύπου hire employee το οποίο σηματοδοτεί την πρόσληψη ενός νέου εργαζόμενου με αναγνωριστκό **eid** στο πάρκο. Ο εργαζόμενος αρχικά έχει κενή λίστα ζώων για τα οποία είναι υπεύθυνος και συνεπώς ο αντίστοιχος μετρητής (**animal_count**) έχει τιμή 0. Ο νέος κόμβος θα πρέπει να εισάγεται στο δένδρο εργαζομένων σε χρόνο **O(h)**, όπου h είναι το ύψος του δένδρου εργαζομένων, εξασφαλίζοντας την ιδιότητα της ταξινόμησής του. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
H <eid>
      EMPLOYEES: <eid_1>, <eid_2>, … <eid_n>
DONE
```

όπου n είναι το μέγεθος του δένδρου εργαζομένων και για κάθε i, $1 \le i \le n$, $<eid_i>$, είναι το αναγνωριστικό του i-οστού εργαζομένου, όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του δένδρου αυτού. Οι κόμβοι θα πρέπει να έχουν εισαχθεί με τέτοιο τρόπο ώστε αν πραγματοποιηθεί ενδοδιατεταγμένη (in-order) διάσχιση στο δένδρο, οι εργαζόμενοι να προσπελάζονται σε αύξουσα διάταξη ως προς το πεδίο eid.

- A <eid> <aid>

Γεγονός τύπου assign animal σε εργαζόμενο το οποίο σηματοδοτεί την ανάθεση του ζώου με αναγνωριστικό **aid** στον εργαζόμενο με αναγνωριστικό **eid**. Το ζώο με αναγνωριστικό **aid** προστίθεται στη λίστα των ζώων για τα οποία ο συγκεκριμένος εργαζόμενος είναι υπεύθυνος και αυξάνεται το πεδίο **animal_count** του εργαζομένου. Ακόμη, πρέπει να γίνει και η κατάλληλη εισαγωγή στον πίνακα κατακερματισμού με βάση το αναγνωριστικό του ζώου. Η κατάλληλη θέση του πίνακα θα βρίσκεται αφού εφαρμοστεί η συνάρτηση κατακερματισμού στο αναγνωριστικό του ζώου.

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

```
A <eid> <aid>
      EMPLOYEES:
      <employee_1: animal_count_1> : <aid_1,1> . . . <aid_1,n1>
      <employee_2: animal_count_2> : <aid_2,1> . . . <aid_2,n2>
      . . .
      <employee_k: animal_count_k> : <aid_k,1> . . . <aid_k,nk>
DONE
```

όπου k είναι το μέγεθος του δένδρου εργαζομένων, για κάθε i, $1 \le i \le k$, n_k είναι το μέγεθος της λίστας ζώων του i-οστού εργαζομένου και για κάθε j, $1 \le j \le n_k$, και $<aid_{i,j}>$ είναι το αναγνωριστικό του j-οστού ζώου στην λίστα ζώων του i-οστού εργαζομένου του δένδρου, όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του.

- R <eid>

Γεγονός τύπου retirement εργαζομένου το οποίο σηματοδοτεί τη συνταξιοδότηση του εργαζομένου με αναγνωριστικό **eid**. Κατά το γεγονός αυτό πρέπει να διαγράψετε από το δένδρο εργαζομένων τον κόμβο που αντιστοιχεί στον συγκεκριμένο εργαζόμενο. Ο κόμβος αυτός θα πρέπει να εντοπίζεται σε χρόνο **O(h).** Τα ζώα για τα οποία είναι υπεύθυνος ο εργαζόμενος πρέπει να ανατεθούν σε κάποιον άλλον εργαζόμενο ώστε να συνεχιστεί η ομαλή λειτουργία του πάρκου. Πιο συγκεκριμένα, τα ζώα του εργαζομένου που συνταξιοδοτείται θα μεταφέρονται στον **αμέσως προηγούμενο** κόμβο σύμφωνα με την **ενδοδιατεταγμένη** (in-order) διάσχιση

του δένδρου εργαζομένων. Σε περίπτωση που ο εργαζόμενος δεν έχει προηγούμενο κόμβο κατά την ενδοδιατεταγμένη διάσχιση, τότε τα ζώα για τα οποία είναι υπεύθυνος θα πρέπει να μεταφέρονται στον δεξιότερο (rightmost) κόμβο του δένδρου (δηλαδή εκείνον που συναντάται τελευταίος στην ενδοδιατεταγμένη διάσχιση του δένδρου). Η διαδικασία εντοπισμού του νέου εργαζόμενου που θα αναλάβει τα ζώα θα πρέπει να εκτελείται σε χρόνο **O(d)**, όπου d είναι το ύψος του κόμβου που διαγράφεται.

Αξιοσημείωτο είναι ότι κατά την μεταφορά των ζώων από τον εργαζόμενο που συνταξιοδοτείται, όταν τα ζώα μεταφερθούν στην λίστα ζώων του κατάλληλου εργαζομένου θα πρέπει να ανανεώνεται και η αντίστοιχη εγγραφή στον πίνακα κατακερματισμού για το κάθε ζώο. Συγκεκριμένα, θα πρέπει να εντοπίζετε την εγγραφή στον πίνακα κατακερματισμού με βάση το αναγνωριστικό του ζώου που μεταφέρεται και να ανανεώνετε την τιμή του **eid** σε αυτήν του καινούργιου εργαζόμενου που το ανέλαβε.

Για τα γεγονότα αυτά μπορείτε να κάνετε την παραδοχή ότι το δένδρο εργαζομένων θα περιέχει σίγουρα τουλάχιστον δύο εργαζομένους.

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

```
R <eid>
      EMPLOYEES:
      <employee_1: animal_count_1> : <aid_1,1> . . . <aid_1,n1>
      <employee_2: animal_count_2> : <aid_2,1> . . . <aid_2,n2>
      . . .
      <employee_k: animal_count_k> : <aid_k,1> . . . <aid_k,nk>
DONE
```

όπου k είναι το μέγεθος του δένδρου εργαζομένων, για κάθε i, $1 \le i \le k$, n_k είναι το μέγεθος της λίστας ζώων του i-οστού εργαζομένου και για κάθε j, $1 \le j \le n_k$, και $<aid_{i,j}>$ είναι το αναγνωριστικό του j-οστού ζώου στην λίστα ζώων του i-οστού εργαζομένου του δένδρου, όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του.

- X

Γεγονός τύπου print ecosystems, κατά το οποίο θα πρέπει να τυπώσετε όλα τα οικοσυστήματα του πίνακα οικοσυστημάτων. Για το κάθε οικοσύστημα θα πρέπει να τυπώνονται όλες οι πληροφορίες του συμπεριλαμβανομένου και του δένδρου ζώων του οικοσυστήματος. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
X

    ECOSYSTEMS:

    <ecosystem_1>: <aid_1,1> . . . <aid_1,n1>
    <ecosystem_2>: <aid_2,1> . . . <aid_2,n2>
    ...
   <ecosystem_50>: <aid_50,1> . . . <aid_50,n50>
DONE
```

όπου για κάθε i, $1 \le i \le 50$, n_i είναι το μέγεθος του δένδρου ζώων του i-οστού οικοσυστήματος του πίνακα οικοσυστημάτων, και για κάθε j, $1 \le j \le n_i$, και $<aid_{i,j}>$ είναι το αναγνωριστικό του j-οστού κόμβου στο δένδρο ζώων του i-οστού οικοσυστήματος, όπως προκύπτει από την ενδοδιατεταγμένη διάσχιση του δένδρου αυτού.

- Y

Γεγονός τύπου print visitors, κατά το οποίο θα πρέπει να τυπώσεττε το πλήρες δένδρο των επισκεπτών του πάρκου. Η διάσχιση κι εκτύπωση του δένδρου θα πρέπει να γίνει κι εδώ ανά επίπεδα με την μέθοδο που περιγράφηκε παραπάνω στο γεγονός **V**. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα

```
Y
Level0:          <vid0,1 : visits0,1 : last_visit0,1>
Level1:          <vid1,1 : visits1,1 : last_visit1,1>, <vid1,2 : visits1,2 : last_visit1,2>
.
.
Levelh:          <vidh,1 : visitsh,1 : last_visith,1> … <vidh,n : visitsh,n : last_visith,n>
DONE
```

πρέπει να τυπώνει την ακόλουθη πληροφορία:

όπου για κάθε i, $0 < i < h$, h το ύψος του δένδρου, n_i ο αριθμός των κόμβων στο επίπεδο i και για κάθε j, $1 \leq j \leq n_i$, και $<vid_{i,j} : visits_{i,j} : last\_visit_{i,j}>$ τα πεδία του j-οστού κόμβου στο επίπεδο i του δένδρου.

- Z

Γεγονός τύπου print employees, κατά το οποίο θα πρέπει να τυπώσετε το δένδρο των εργαζομένων. Για τον κάθε εργαζόμενο θα πρέπει να τυπώνονται όλες οι πληροφορίες του συμπεριλαμβανομένης και της λίστας ζώων του. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
Z
      EMPLOYEES:
      <employee1: animal_count1> : <aid1,1> . . . <aid1,n1>
      <employee2: animal_count2> : <aid2,1> . . . <aid2,n2>
      . . .
      <employeek: animal_countk> : <aidk,1> . . . <aidk,nk>
DONE
```

όπου k είναι το μέγεθος του δένδρου εργαζομένων, για κάθε i, $1 \leq i \leq k$, n_k είναι το μέγεθος της λίστας ζώων του i-οστού εργαζομένου και για κάθε j, $1 \leq j \leq n_k$, και $<aid_{i,j}>$ είναι το αναγνωριστικό του j-οστού ζώου στην λίστα ζώων του i-οστού εργαζομένου του δένδρου εργαζομένων, όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του.

-E

Γεγονός τύπου print animal to employees, κατά το οποίο θα πρέπει να τυπώσετε τον πίνακα κατακερματισμού που αντιστοιχίζει τα ζώα με τους υπεύθυνους για αυτά εργαζομένους. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
E
Chain <0>:
      <aid0,1> : <eid0,1>
      <aid0,2> : <eid0,2>
      …
      <aid0,n0> : <eid0,n0>
…
Chain <j>:
      <aidj,1> : <eidj,1>
      <aidj,2> : <eidj,2>
      …
      <aidj,nj> : <eidj,nj>
…
DONE
```

όπου για κάθε j είναι η j-οστή αλυσίδα του πίνακα κατακερματισμού και n_j είναι ο αριθμός των animal_to_employee structs που έχουν τοποθετηθεί στην αλυσίδας αυτή και για κάθε i, $1 \le i \le n_j$, και $<aid_{j,i}>$ και $<eid_{j,i}>$ είναι τα αναγνωριστικά του ζώου και του εργαζομένου αντίστοιχα.

Bonus:

Όσοι φοιτητές επιθυμούν, μπορούν να παραδώσουν και μια δεύτερη υλοποίηση στην οποία τα δένδρα του κάθε οικοσυστήματος στον πίνακα οικοσυστημάτων θα είναι τύπου **AVL**. Η υλοποίηση με την χρήση AVL δένδρων δεν είναι υποχρεωτική, αλλά θα είναι ένα επιπρόσθετο bonus στην βαθμολογία σας. Παραταύτα η υλοποίηση σύμφωνα με την εκτενή περιγραφή που δόθηκε παραπάνω είναι **υποχρεωτική και θα πρέπει να παραδοθεί σε κάθε περίπτωση**.

Είναι αξιοσημείωτο ότι στην υλοποίηση με AVL δένδρα οι κόμβοι του κάθε AVL δένδρου θα έχουν δείκτες και προς του πατρικούς τους κόμβους, τους οποίους θα πρέπει να προσθέσετε εσείς στο κατάλληλο struct στο αρχείο zoo.h στον φάκελο που θα παραδώσετε για αυτήν την υλοποίηση. Επίσης, το γεγονός **D** θα υλοποιείται διαφορετικά. Συγκεκριμένα, θα πρέπει να διασχίζετε το δένδρο ζώων και κάθε ζώο που συναντάτε θα πρέπει να το εισάγετε στο κατάλληλο AVL δένδρο του οικοσυστήματος στο οποίο ανήκει. Μετά το πέρας της εκτέλεσης του γεγονότος **D** το αρχικό δένδρο ζώων θα πρέπει να είναι κενό και η πληροφορία που θα τυπώνεται θα είναι η ίδια με το γεγονός **D** που περιγράφηκε παραπάνω.

Όσοι φοιτητές επιλέξουν να υλοποιήσουν και το bonus θα πρέπει να παραδώσουν **δύο (2)** φακέλους, έναν για κάθε εκδοχή των δένδρων. Ο κάθε φάκελος θα πρέπει να περιέχει οτιδήποτε χρειάζεται για να κάνει compile και να εκτελείται η εργασία.

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 animal {
        int aid;  // the animal id
```

```
        struct animal *lc; // pointer to the node's left child
        struct animal *rc; // pointer to the node's right child
};


struct ecosystem {
        int eco_id;  // the ecosystem's id
        struct animal * animals_tree;  // pointer to the ecosystem's animals tree
};

struct visitor {
        int vid;  // the visitor's id
        int last_visit;  // the year the visitor last visited the park
        int visits;  // the total number of visits the visitor has made
        struct visitor * p;  // pointer to the node's parent
        struct visitor * lc; // pointer to the node's left child
        struct visitor * rc; // pointer to the node's right child
};

struct employee {
        int eid;  // the employee's id
        int animal_count;  // the number of animals the employee is responsible for
        struct employee_animal * employees_animal_list;  // pointer to the employee's animal list
        struct employee * lc; // pointer to the node's left child
        struct employee * rc; // pointer to the node's right child
}

struct employee_animal{
        int aid;  // the animal's id
        struct employee_animal * next;  // pointer to the next node of the list
}

struct animal_to_employee {
        int aid;  // the animal's id
        int eid; // the id of the employee who is responsible for the animal
        struct animal_to_employee * next; // pointer to the next node of the list
```

```
}


/*The array of the park's ecosystems */
struct ecosystem ecosystems_array [50];
/* global variable, pointer to the root of the animals tree*/
struct animal * animals_tree;
/* global variable, pointer to the root of the visitors tree*/
struct visitor * visitors_tree;
/*global variable, pointer to the rightmost leaf of the visitors tree*/
struct visitor * rightmost_visitor;
/* global variable, pointer to the root of the employees tree*/
struct employee * employees_tree;
/* global variable, pointer to the sentinel node of the employees tree*/
struct employee * employees_sentinel;

/*The hash table mapping animal ids to employees*/
struct * animal_to_employee ANIMALS_TO_EMPLOYEES[hash_table_size];
int animals_total_count;  // The maximum number of animals that are going to live in the park
int max_aid;  // The maximum animal id
int primes[160];  // Array of primes for hashing
```