ΗΥ240: Δομές Δεδομένων
Χειμερινό Εξάμηνο - Ακαδημαϊκό Έτος 2024- 25
Διδάσκουσα: Παναγιώτα Φατούρου

Προγραμματιστική Εργασία - 2ο Μέρος

Ημερομηνία Παράδοσης: Παρασκευή, 20 Δεκεμβρίου 2024, ώρα 23:59.

Τρόπος Παράδοσης: Μέσω της σελίδας elearn του μαθήματος. Πληροφορίες σχετικά με το elearn παρέχονται στην ιστοσελίδα του μαθήματος.

Γενική Περιγραφή

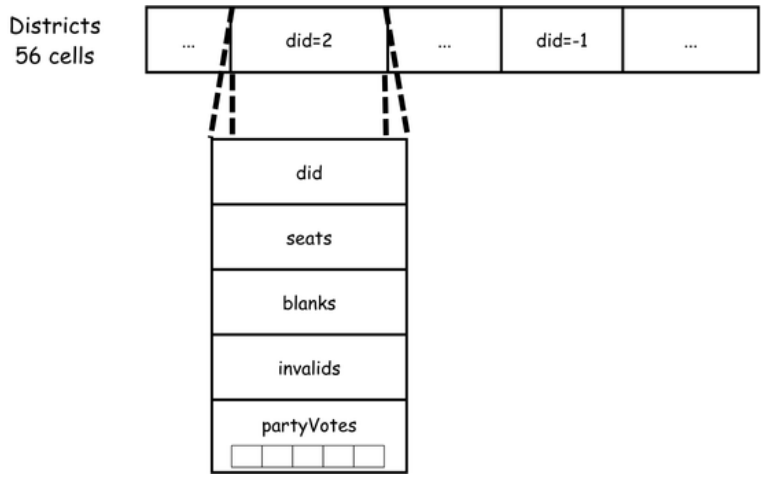
Στην εργασία αυτή καλείστε να υλοποιήσετε ένα πρόγραμμα που προσομοιώνει το εκλογικό σύστημα της Ελλάδας. Το **ελληνικό εκλογικό σύστημα** είναι το σύστημα με το οποίο κατανέμονται οι βουλευτικές έδρες στα κόμματα με βάση τις ψήφους που έλαβαν στις βουλευτικές εκλογές. Ρυθμίζεται από ειδικό νόμο, που ονομάζεται εκλογικός νόμος.

Σημείωση: Προκειμένου να πραγματωθούν οι εκπαιδευτικοί στόχοι της εργασίας, δεν ακολουθείται πιστά ο εκλογικός νόμος και έχουν γίνει σημαντικές απλοποιήσεις σχετικά με τον τρόπο με τον οποίο κατανέμονται οι έδρες. Συνεπώς, όσα αναγράφονται στην παρούσα εκφώνηση δεν αντιστοιχούν με ακρίβεια στην ισχύουσα εκλογική νομοθεσία.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Για την υλοποίηση της προσομοίωσης του ελληνικού εκλογικού συστήματος, θα πρέπει να χρησιμοποιήσετε έναν πίνακα σταθερής χωρητικότητας **56 θέσεων**, ο οποίος ονομάζεται **πίνακας εκλογικών περιφερειών (Districts)** (Σχήμα 1). Κάθε στοιχείο του πίνακα εκλογικών περιφερειών (Districts) είναι μια εγγραφή τύπου **District** με τα ακόλουθα πεδία:

- **did:** Αναγνωριστικό (int) που χαρακτηρίζει μοναδικά την περιφέρεια.
- **seats:** Αριθμός (int) των βουλευτικών εδρών που αντιστοιχούν στην περιφέρεια.
- **blanks:** Αριθμός (int) των **λευκών** ψήφων που μετρήθηκαν κατά την καταμέτρηση ψήφων σε όλα τα εκλογικά τμήματα της περιφέρειας.
- **invalids:** Αριθμός (int) των **άκυρων** ψήφων που μετρήθηκαν κατά την καταμέτρηση ψήφων σε όλα τα εκλογικά τμήματα της περιφέρειας.
- **partyVotes:** Πίνακας (int[]) μεγέθους ίσο με τον αριθμό των κομμάτων που συμμετέχουν στις εκλογές (σ' αυτήν την εργασία θα θεωρήσετε πως το μέγεθος αυτό είναι 5). Τα αναγνωριστικά κομμάτων (pid) χρησιμοποιούνται για τη διευθυνσιοδότηση του πίνακα. Στο στοιχείο partyVotes[j], όπου $0 \leq j \leq 4$, αποθηκεύεται ο αριθμός των ψήφων που έχει λάβει το κόμμα με αναγνωριστικό j στην εκλογική περιφέρεια με αναγνωριστικό did.



Σχήμα 1 Πίνακας εκλογικών περιφερειών (Districts).

Για την αποθήκευση των εκλογικών τμημάτων όλων των περιφερειών θα πρέπει να υλοποιήσετε έναν πίνακα κατακερματισμού, ο οποίος ονομάζεται **πίνακας κατακερματισμού εκλογικών τμημάτων (Stations)** (Σχήμα 2). Επομένως, στη δεύτερη φάση της προγραμματιστικής εργασίας, δεν υπάρχει πλέον μια λίστα εκλογικών τμημάτων ανά περιφέρεια, αλλά μια συγκεντρωτική δομή, που είναι ένας πίνακας κατακερματισμού και αποθηκεύει πληροφορίες για όλα τα εκλογικά τμήματα ανεξάρτητα από την εκλογική περιφέρεια (District) στην οποία αυτά ανήκουν. Για την επίλυση των συγκρούσεων θα πρέπει να ακολουθήσετε τη μέθοδο των ταξινομημένων αλυσίδων. Επομένως, σε κάθε κελί του πίνακα κατακερματισμού εκλογικών τμημάτων (Stations) αποθηκεύεται μια αλυσίδα, κάθε στοιχείο της οποίας αντιστοιχεί σε ένα **εκλογικό τμήμα (Station)**. Κάθε εγγραφή τύπου **Station** αποτελείται από τα παρακάτω πεδία:

- **sid:** Αναγνωριστικό (int) που χαρακτηρίζει μοναδικά το εκλογικό τμήμα.
- **did:** Αναγνωριστικό (int) της εκλογικής περιφέρειας (District) στην οποία ανήκει το εκλογικό τμήμα.

- **registered:** Αριθμός (int) των ψηφοφόρων που είναι εγγεγραμμένοι στο εκλογικό τμήμα.
- **voters:** Δείκτης στη ρίζα του δένδρου ψηφοφόρων του εκλογικού τμήματος με αναγνωριστικό sid (το οποίο περιγράφεται πιο αναλυτικά στη συνέχεια. Κάθε κόμβος αυτού του δένδρου είναι μια εγγραφή (struct) τύπου Voter.
- **next:** Δείκτης (Station*) στον επόμενο κόμβο της αλυσίδας που ανήκει το εκλογικό τμήμα με αναγνωριστικό sid.

Η ταξινόμηση των στοιχείων μιας ταξινομημένης αλυσίδας γίνεται βάση των αναγνωριστικών των εκλογικών τμημάτων (sid) της αλυσίδας.

Τη χωρητικότητα του πίνακα κατακερματισμού εκλογικών τμημάτων την επιλέγετε εσείς και θα πρέπει να είστε σε θέση να δικαιολογήσετε την επιλογή σας. Για την υλοποίηση της συνάρτησης κατακερματισμού θα πρέπει να χρησιμοποιήσετε τη μέθοδο του καθολικού κατακερματισμού.

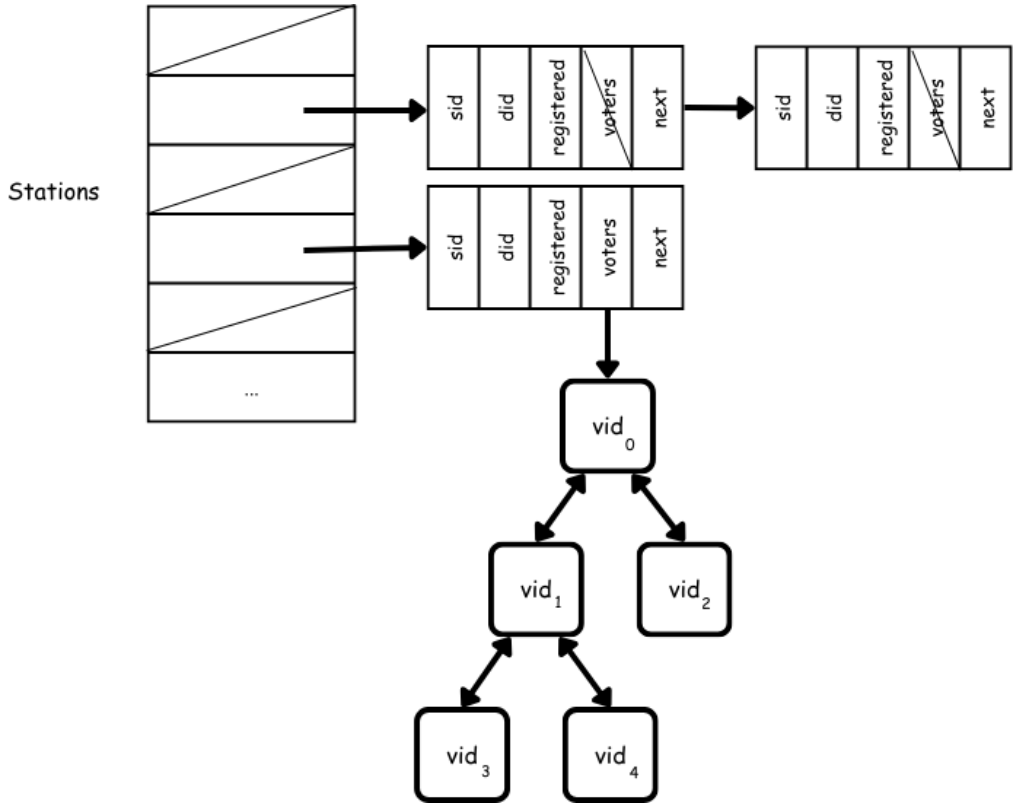
Για την επιλογή της χωρητικότητάς του πίνακα και την υλοποίηση του καθολικού κατακερματισμού δίνονται τα εξής:

- **Primes:** Πίνακας (int[]) πρώτων αριθμών σε αύξουσα σειρά.
- **MaxStationsCount:** Αριθμός (int) που αναπαριστά το μέγιστο πλήθος των εκλογικών τμημάτων.
- **MaxSid:** Αριθμός (int) που αναπαριστά το μέγιστο αναγνωριστικό εκλογικού τμήματος (sid).

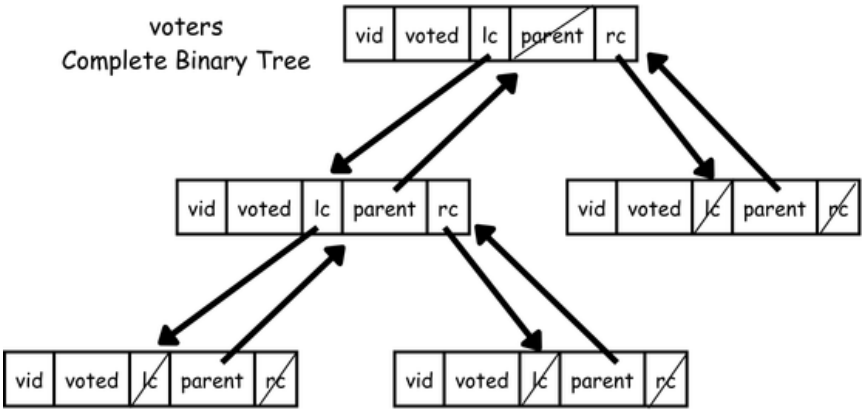
Αυτές οι μεταβλητές είναι **global**, έχουν δηλωθεί στα voting.h, voting.c και αρχικοποιούνται στη main, βάσει τιμών που αναγράφονται στις πρώτες γραμμές κάθε testfile.

Το **δένδρο ψηφοφόρων (voters)** του εκλογικού τμήματος με αναγνωριστικό sid (που αναφέρθηκε παραπάνω) είναι **ένα πλήρες, μη-ταξινομημένο, διπλά-συνδεδεμένο δυαδικό δένδρο** (Σχήμα 3). Κάθε στοιχείο αυτού του δένδρου αντιστοιχεί σε έναν ψηφοφόρο που είναι εγγεγραμμένος στο εκλογικό τμήμα με αναγνωριστικό sid και αποτελεί μια εγγραφή τύπου **Voter** με τα ακόλουθα πεδία:

- **vid:** Αναγνωριστικό (int) που χαρακτηρίζει μοναδικά τον ψηφοφόρο.
- **voted:** Μεταβλητή (bool) που δηλώνει αν ο ψηφοφόρος έχει εξασκήσει το δικαίωμα ψήφου.
- **parent:** Δείκτης (Voter*) στον πατρικό κόμβο.
- **lc:** Δείκτης (Voter*) στο αριστερό παιδί του κόμβου.
- **rc:** Δείκτης (Voter*) στο δεξί παιδί του κόμβου.



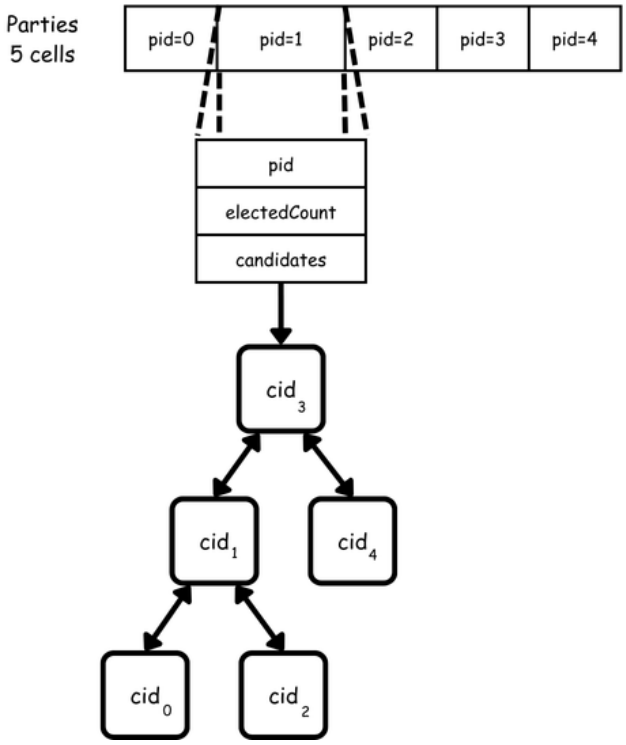
Σχήμα 2 Πίνακας κατακερματισμού εκλογικών τμημάτων (Stations).



Σχήμα 3 Δένδρο ψηφοφόρων (voters) πλήρες, μη-ταξινομημένο, διπλά-συνδεδεμένο δυαδικό δένδρο.

Για τις ανάγκες της εργασίας αυτής, θεωρούμε ότι υπάρχουν πέντε διαφορετικά κόμματα που συμμετέχουν στις εκλογές. Θα πρέπει να χρησιμοποιήσετε έναν πίνακα σταθερής χωρητικότητας **5 θέσεων**, ο οποίος ονομάζεται **πίνακας κομμάτων (Parties)** (Σχήμα 4). Κάθε στοιχείο του πίνακα κομμάτων είναι μια εγγραφή τύπου **Party** με τα ακόλουθα πεδία:

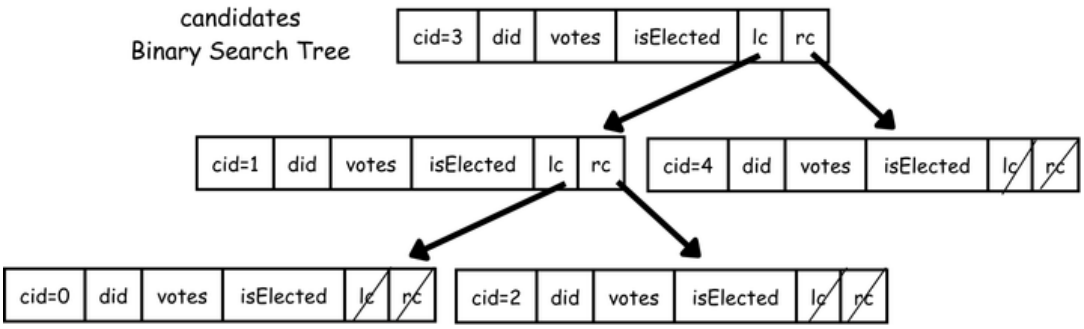
- **pid:** Αναγνωριστικό (int) που χαρακτηρίζει μοναδικά το κόμμα.
- **electedCount:** Αριθμός (int) των βουλευτών που εκλέγει το κόμμα.
- **candidates:** Δείκτης στη ρίζα του δένδρου υποψήφιων του κόμματος με αναγνωριστικό pid (το οποίο περιγράφεται πιο αναλυτικά στη συνέχεια). Κάθε κόμβος αυτού του δένδρου είναι μια εγγραφή (struct) τύπου Candidate.



Σχήμα 4 Πίνακας κομμάτων (Parties).

Το **δένδρο υποψήφιων (candidates)** του κόμματος αποθηκεύει πληροφορίες για τους υποψήφιους βουλευτές ενός κόμματος. Είναι ένα **απλά συνδεδεμένο δένδρο δυαδικής αναζήτησης (Binary Search Tree), ταξινομημένο** βάσει του αναγνωριστικού του υποψηφίου (cid) (Σχήμα 5). Κάθε στοιχείο του δένδρο υποψήφιων (candidates) κόμματος pid, αποτελεί μια εγγραφή τύπου **Candidate** και αντιστοιχεί σε έναν υποψήφιο βουλευτή του κόμματος με αναγνωριστικό pid. Κάθε εγγραφή τύπου Candidate αποτελείται από τα παρακάτω πεδία :

- **cid:** Αναγνωριστικό (int) που χαρακτηρίζει μοναδικά τον υποψήφιο βουλευτή.
- **did:** Αναγνωριστικό (int) της εκλογικής περιφέρειας (District), στην οποία ο υποψήφιος εκλέγεται βουλευτής.
- **votes:** Αριθμός (int) των ψήφων που συγκέντρωσε ο υποψήφιος από όλα τα εκλογικά τμήματα της εκλογικής περιφέρειας με αναγνωριστικό did στην οποία είναι υποψήφιος.
- **isElected:** Μεταβλητή (bool) που δηλώνει αν ο υποψήφιος έχει εκλεγεί βουλευτής.
- **lc:** Δείκτης (Candidate*) στο αριστερό παιδί του κόμβου.
- **rc:** Δείκτης (Candidate*) στο δεξί παιδί του κόμβου.

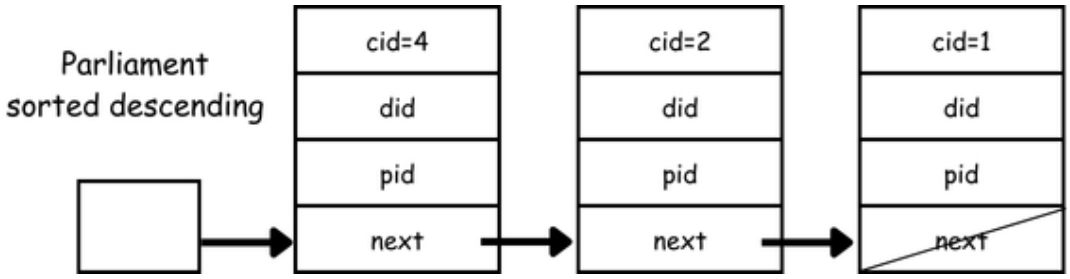


Σχήμα 5 Δένδρο υποψηφίων (candidates) απλά-συνδεδεμένο δυαδικό δένδρο αναζήτησης, ταξινομημένο ως προς το αναγνωριστικό του κάθε υποψηφίου.

Τέλος, θα πρέπει να υλοποιήσετε τη δομή του κοινοβουλίου η οποία περιέχει τους εκλεγμένους βουλευτές. Ονομάζεται **κοινοβούλιο (Parliament)** και είναι μια **ταξινομημένη, απλά συνδεδεμένη λίστα, σε φθίνουσα διάταξη ως προς το αναγνωριστικό του βουλευτή (cid)** (Σχήμα 6). Η μεταβλητή Parliament είναι ένας δείκτης (ElectedCandidate*) στο πρώτο στοιχείο της λίστας. Κάθε κόμβος αυτής της λίστας αποτελεί μια εγγραφή τύπου **ElectedCandidate** με πεδία:

- **cid:** Αναγνωριστικό (int) που χαρακτηρίζει μοναδικά τον υποψήφιο που έχει εκλεγεί βουλευτής.
- **did:** Αναγνωριστικό (int) της εκλογικής περιφέρειας (District) στην οποία ο βουλευτής έχει εκλεγεί.
- **pid:** Αναγνωριστικό (int) που αντιστοιχεί στο κόμμα (Party) το οποίο ο βουλευτής εκπροσωπεί.
- **next:** Δείκτης (ElectedCandidates*) στον επόμενο κόμβο της λίστας

Η δομή του **κοινοβουλίου (Parliament)** δεν περιέχει αρχικά στοιχεία.



Σχήμα 6 Κοινοβούλιο (Parliament) απλά συνδεδεμένη λίστα, ταξινομημένη σε φθίνουσα σειρά.

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:

– **A <MaxStationsCount> <MaxSid>**

Γεγονός τύπου *announce elections* το οποίο υποδηλώνει την αναγγελία των εκλογών. Κατά το γεγονός αυτό αρχικοποιούνται οι δομές:

- πίνακας εκλογικών περιφερειών (Districts)
- πίνακας κατακερματισμού εκλογικών τμημάτων (Stations)
- πίνακας κομμάτων (Parties)
- κοινοβούλιο (Parliament)

Οι αρχικές τιμές των παραπάνω δομών περιγράφονται συγκεντρωτικά στην Ενότητα «Αρχικές τιμές πεδίων δομών δεδομένων» στο τέλος της εκφώνησης της εργασίας.

Επίσης αρχικοποιούνται οι global μεταβλητές MaxStationsCount με <MaxStationsCount>, και MaxSid με <MaxSid>.

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

```
A <MaxStationsCount> <MaxSid>
DONE
```

– **D <did> <seats>**

Γεγονός τύπου *create district* το οποίο σηματοδοτεί τη δημιουργία μιας νέας εκλογικής περιφέρειας (District) με αναγνωριστικό <did> και αριθμό εκλεγόμενων εδρών <seats>. Η νέα εκλογική περιφέρεια αρχικοποιείται κατάλληλα και εισάγεται στον πίνακα εκλογικών περιφερειών. **Η χρονική πολυπλοκότητα της εισαγωγής πρέπει να είναι O(log n)** (επομένως θα πρέπει να υλοποιήσετε αλγόριθμο ώστε να βρίσκεται το πρώτο κενό κελί του πίνακα Districts σε O(log n) χρόνο). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
D <did> <seats>
  Districts
  <did₁>, <did₂>, ..., <didₙ>
DONE
```

όπου n είναι ο αριθμός των εκλογικών περιφερειών που έχουν δημιουργηθεί μέχρι τώρα και για κάθε i ∈ {1, ..., n}, $<did_i>$ είναι το αναγνωριστικό της i-οστής εκλογικής περιφέρειας.

– **S <sid> <did>**

Γεγονός τύπου *create station* το οποίο σηματοδοτεί τη δημιουργία ενός νέου εκλογικού τμήματος (Station) με αναγνωριστικό <sid> στην εκλογική περιφέρεια με αναγνωριστικό <did>. Η εισαγωγή του εκλογικού τμήματος γίνεται στην κατάλληλη αλυσίδα του πίνακα κατακερματισμού, βάσει της τιμής κατακερματισμού του αναγνωριστικού του <sid>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
S <sid> <did>
Stations[h]
<sid₁>, <sid₂>, ..., <sidₙ>
DONE
```

όπου h είναι η τιμή κατακερματισμού του <sid>, n είναι ο αριθμός των εκλογικών τμημάτων της συγκεκριμένης αλυσίδας και για κάθε i ∈ {1, …, n} <sid_i> είναι το αναγνωριστικό του i -οστού εκλογικού τμήματος στην αλυσίδα αυτή.

– **R <vid> <sid>**

Γεγονός τύπου *register voter* το οποίο σηματοδοτεί την εγγραφή ενός νέου ψηφοφόρου (Voter) με αναγνωριστικό <vid> στο εκλογικό τμήμα με αναγνωριστικό <sid>. Το γεγονός αυτό προσθέτει το νέο ψηφοφόρο στο δένδρο ψηφοφόρων του εκλογικού τμήματος με αναγνωριστικό <sid>.

Προσοχή: Η εισαγωγή θα πρέπει να γίνεται έτσι ώστε το δένδρο να παραμένει πλήρες (δηλαδή με τρόπο συμβατό με την **Άσκηση 3 της 3ης Σειράς Ασκήσεων**). Η χρονική πολυπλοκότητα της λειτουργίας αυτής πρέπει να είναι O(log n), όπου n είναι το πλήθος των στοιχείων στο δένδρο.

Επιπλέον σε αυτό το σημείο αυξάνεται κατά ένα το πλήθος (πεδίο registered του struct Station) των εγγεγραμμένων ψηφοφόρων του εκλογικού τμήματος με αναγνωριστικό <sid>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
R <vid> <sid>
  Voters[<sid>]
  <vid₁>, <vid₂>, ..., <vidₙ>
DONE
```

όπου n είναι το πλήθος των ψηφοφόρων του εκλογικού τμήματος με αναγνωριστικό <sid> και για κάθε i ∈ {1, …, n}, <vid_i> είναι το αναγνωριστικό του i-οστού ψηφοφόρου στο δένδρο ψηφοφόρων του εκλογικού τμήματος με αναγνωριστικό <sid>, στη διάταξη που προκύπτει όταν εφαρμόζεται ενδοδιατεταγμένη διάσχιση στο δένδρο ψηφοφόρων του εκλογικού τμήματος με αναγνωριστικό <sid>.

– **C <cid> <pid> <did>**

Γεγονός τύπου *register candidate* το οποίο σηματοδοτεί την εγγραφή ενός νέου υποψηφίου (Candidate) με αναγνωριστικό <cid> του κόμματος με αναγνωριστικό <pid> στην εκλογική περιφέρεια με αναγνωριστικό <did>. Το γεγονός αυτό προσθέτει το νέο υποψήφιο στο δένδρο υποψήφιων του κόμματος με αναγνωριστικό <pid>. Η χρονική πολυπλοκότητα της εισαγωγής πρέπει να είναι O(h), όπου h είναι το ύψος του δέντρου.

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

```
C <cid> <pid> <did>
  Candidates[<pid>]
  <cid₁> <did₁>,
  <cid₂> <did₂>,
  …,
  <cidₙ> <didₙ>
DONE
```

όπου <pid> είναι το αναγνωριστικό του κόμματος που ανήκει ο υποψήφιος, n είναι το πλήθος των υποψηφίων στο δένδρο υποψήφιων του κόμματος <pid> και για κάθε i ∈ {1, …, n}:

- <cid_i> είναι το αναγνωριστικό του i-οστού υποψηφίου (σύμφωνα με την ενδοδιατεταγμένη διάσχιση) στο δένδρο υποψήφιων του κόμματος <pid>,
- <did_i> είναι το αναγνωριστικό της περιφέρειας στην οποία εκλέγεται ο υποψήφιος με αναγνωριστικό <cid_i>

Προσοχή: Για την εκτύπωση των αναγνωριστικών των υποψηφίων πρέπει να εφαρμόζεται ενδοδιατεταγμένη διάσχιση στο δένδρο υποψήφιων του κόμματος <pid>.

– **V <vid> <sid> <cid> <pid>**

Γεγονός τύπου *vote* το οποίο σηματοδοτεί τη διαδικασία ψήφου ενός ψηφοφόρου . Ο ψηφοφόρος (Voter) με αναγνωριστικό <vid> που ανήκει στο εκλογικό τμήμα (Station) με αναγνωριστικό <sid> ψηφίζει τον υποψήφιο (Candidate) με αναγνωριστικό <cid> (ή λευκό, ή άκυρο) που ανήκει στο κόμμα (Party) με αναγνωριστικό <pid>.

Το γεγονός αυτό εντοπίζει πρώτα το εκλογικό τμήμα με αναγνωριστικό <sid> στον πίνακα κατακερματισμού εκλογικών τμημάτων. Στη συνέχεια εντοπίζει τον ψηφοφόρο με αναγνωριστικό <vid> στο δένδρο ψηφοφόρων του εκλογικού του τμήματος και σημειώνει ότι συμμετείχε στις εκλογές (θέτοντας σε true το πεδίο voted του struct Voter). Από το struct που αντιστοιχεί στον ψηφοφόρο μπορεί να γίνει γνωστή η περιφέρεια <did> στην οποία ανήκει το εκλογικό τμήμα του ψηφοφόρου.

Σε αυτό το σημείο προσμετράτε και η ψήφος ως εξής:

- Αν το <cid> είναι ίσο με το δεσμευμένο αναγνωριστικό -1, τότε αυξάνεται κατά ένα ο αριθμός των λευκών της περιφέρειας (πεδίο blanks) στην οποία ανήκει το εκλογικό τμήμα του ψηφοφόρου.
- Ομοίως, αν το <cid> είναι ίσο με το δεσμευμένο αναγνωριστικό -2, τότε αυξάνεται κατά ένα ο αριθμός των άκυρων της περιφέρειας (πεδίο invalids) στην οποία ανήκει το εκλογικό τμήμα του ψηφοφόρου.

- Σε κάθε άλλη περίπτωση εντοπίζεται ο υποψήφιος με αναγνωριστικό <cid> στο δένδρο υποψηφίων του κόμματος <pid> και αυξάνεται ο αριθμός των ψήφων του (πεδίο votes στο struct Candidate). Επιπλέον, θα πρέπει να αυξηθεί κατά ένα, ο μετρητής partyVotes[pid] στο struct της εκλογικής περιφέρειας <did> (στην οποία ανήκει το εκλογικό τμήμα του υποψήφιου).

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

```
V <vid> <sid> <cid> <pid>
  District[<did>]
  blanks <blanks>
  invalids <invalids>
  partyVotes
  <pid1> <votes1>,
  <pid2> <votes2>,
  …,
  <pidk> <votesk>
DONE
```

όπου <did> είναι το αναγνωριστικό της εκλογικής περιφέρειας στην οποία εκλέγεται ο υποψήφιος με αναγνωριστικό <cid>, k είναι ο αριθμός των κομμάτων που συμμετέχουν στις εκλογές και για κάθε i ∈ {1, …, k}, <pid_i> είναι το αναγνωριστικό του i-οστού κόμματος και <$votes_i$> το πλήθος των ψήφων που έλαβε το κόμμα με αναγνωριστικό <pid_i> στην εκλογική περιφέρεια, στην οποία ανήκει ο υποψήφιος με αναγνωριστικό <cid>. Επίσης, <blanks> είναι ο αριθμός των λευκών και <invalids> ο αριθμός των άκυρων ψήφων στη συγκεκριμένη εκλογική περιφέρεια.

Εάν το <cid> είναι ένα από τα δεσμευμένα αναγνωριστικά -1 ή -2, τότε το <did> αντιστοιχεί στο αναγνωριστικό της εκλογικής περιφέρειας του ψηφοφόρου με αναγνωριστικό <vid>.

– **M <did>**

Γεγονός τύπου *count votes* το οποίο σηματοδοτεί την καταμέτρηση ψήφων στην εκλογική περιφέρεια με αναγνωριστικό <did>. Στο γεγονός αυτό, θα πρέπει να εκτελείτε τις ακόλουθες ενέργειες για την εκλογική περιφέρεια με αναγνωριστικό <did>:

1. Υπολογίζετε το **εκλογικό μέτρο**, το οποίο ορίζεται ως ο λόγος του συνόλου των **έγκυρων ψήφων** στην εκλογική περιφέρεια <did> (δηλαδή το άθροισμα των στοιχείων του πεδίου partyVotes της εκλογικής περιφέρειας <did>) προς τον αριθμό των εδρών (πεδίο seats του struct District) που της αναλογούν. (Αν ο αριθμός εδρών είναι μηδέν τότε το εκλογικό μέτρο είναι μηδέν).
2. Θα πρέπει να χρησιμοποιήσετε έναν βοηθητικό πίνακα που λέγεται partyElected και έχει πέντε εγγραφές, μια για κάθε κόμμα. Υπολογίζετε πόσοι υποψήφιοι θα εκλεγούν από κάθε κόμμα στην εκλογική περιφέρεια διαιρώντας το σύνολο έγκυρων ψήφων που έλαβε το κόμμα στην περιφέρεια (δηλαδή το Districts[<did>].partyVotes[<pid>]) με το εκλογικό μέτρο, με στρογγυλοποίηση προς τα κάτω. Αυτόν τον αριθμό θα πρέπει να τον αποθηκεύετε στο

partyElected[<pid>] και να τον προσθέτετε στο πεδίο electedCount του αντίστοιχου κόμματος, ενώ ταυτόχρονα τον αφαιρείτε από τις διαθέσιμες έδρες (Districts[<did>].seats) της εκλογικής περιφέρειας. (Αν το εκλογικό μέτρο είναι μηδέν τότε το partyElected[<pid>] είναι μηδέν). Εάν το partyElected[<pid>] είναι μεγαλύτερο από τους διαθέσιμους υποψήφιους του κόμματος <pid>, τότε το θέτετε ίσο με αυτούς (εφαρμόζοντας και τις απαραίτητες διορθώσεις στο πεδίο electedCount του αντίστοιχου κόμματος, και στις διαθέσιμες έδρες της εκλογικής περιφέρειας).

3. Για κάθε κόμμα και για κάθε έδρα που αυτό δικαιούται στην εκλογική περιφέρεια, την έδρα καταλαμβάνει ο υποψήφιος με τις περισσότερες ψήφους στο δένδρο υποψηφίων του κόμματος (ο οποίος πρέπει αρχικά να εντοπιστεί). Για να το πετύχουμε αυτό, θα πρέπει, **για κάθε κόμμα** να εφαρμοστεί ο **Αλγόριθμος εκλογής υποψήφιων κόμματος σε περιφέρεια** (ο οποίος περιγράφεται παρακάτω).

   Εν συντομία, ο αλγόριθμος εκλογής υποψηφίων κόμματος διασχίζει το δένδρο υποψηφίων του κόμματος σε κάθε περιφέρεια και βρίσκει εκείνους που έχουν πάρει τους περισσότερους ψήφους και πρέπει να εκλεγούν. Για να το πετύχει αυτό, χρησιμοποιεί ένα σωρό που ονομάζεται elected (ο πίνακας του σωρού δεσμεύεται δυναμικά με malloc, αποθηκεύει δείκτες σε κόμβους του δένδρου υποψηφίων του κόμματος και έχει μέγεθος partyElected[<pid>]). **Η προτεραιότητα καθορίζεται από τον αριθμό των ψήφων** (πεδίο votes του struct Candidate) που έχουν λάβει οι υποψήφιοι που αποθηκεύονται (έμμεσα) στο σωρό.

Αλγόριθμος εκλογής υποψήφιων κόμματος σε περιφέρεια

ElectPartyCandidatesInDistrict(<pid>, <did>, partyElected[<pid>]) → void

Ο αλγόριθμος εκλογής υποψηφίων κόμματος σε περιφέρεια λαμβάνει εισόδους:

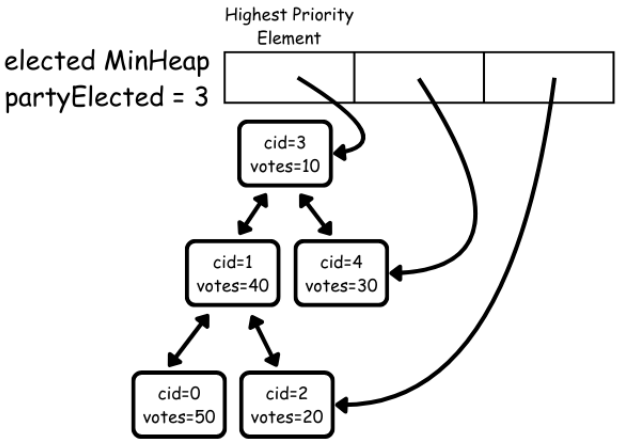
- το αναγνωριστικό κόμματος <pid>
- το αναγνωριστικό περιφέρειας καταμέτρησης ψήφων <did>
- το partyElected[<pid>]

Ο αλγόριθμος δεν επιστρέφει έξοδο και εκτελεί τις ακόλουθες ενέργειες.

1. **Ο πίνακας elected υλοποιεί Σωρό Ελαχίστων (MinHeap), στον οποίο η προτεραιότητα καθορίζεται από τον αριθμό των ψήφων** (πεδίο votes του struct Candidate) που έχουν λάβει οι υποψήφιοι που αποθηκεύονται στον πίνακα (μέσω δεικτών σε αυτούς). Τα κελιά του πίνακα elected αρχικοποιούνται με τους πρώτους υποψηφίους του κόμματος <pid> που πολιτεύονται στην περιφέρεια <did>. Ένα παράδειγμα περιεχομένων του elected μετά την εκτέλεση του αλγορίθμου παρουσιάζεται στο Σχήμα 7.

2. Πραγματοποιείται διάσχιση του δένδρου υποψηφίων του κόμματος <pid> και για κάθε υποψήφιο που πολιτεύεται στην περιφέρεια με αναγνωριστικό <did>, εξετάζεται αν ο αριθμός των ψήφων του είναι μεγαλύτερος από αυτών του υποψηφίου με τις λιγότερες ψήφους στον πίνακα elected . Αν συμβαίνει αυτό τότε αντικαθιστούμε τον υποψήφιο με τις λιγότερες ψήφους στον πίνακα elected με τον τρέχοντα υποψήφιο από το δένδρο υποψηφίων του κόμματος <pid>. Η αντικατάσταση γίνεται έτσι ώστε ο elected να εξακολουθεί να είναι σωρός.

3. Μετά το τέλος της διάσχισης του δένδρου υποψηφίων του κόμματος <pid>, ο πίνακας elected θα περιέχει δείκτες στους κόμβους του δένδρου υποψηφίων του κόμματος <pid> που αντιστοιχούν σε όσους υποψήφιους εκλέγονται στην εκλογική περιφέρεια <did>. Για κάθε έναν από αυτούς τους υποψήφιους, θέτουμε το πεδίο isElected της αντίστοιχης εγγραφής τους στην τιμή true.



Σχήμα 7 Σωρός Ελαχίστων (MinHeap) elected

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

```
M <did>
  seats
  <cid₁> <pid₁> <votes₁>,
  <cid₂> <pid₂> <votes₂>,
  …,
  <cidₙ> <pidₙ> <votesₙ>
DONE
```

όπου n είναι ο αριθμός των εδρών στην εκλογική περιφέρεια με αναγνωριστικό <did> και για κάθε i ∈ {1, …, n}, <cid_i>, <pid_i>, <$votes_i$> είναι το αναγνωριστικό, το κόμμα, και ο αριθμός ψήφων του i-οστού υποψηφίου που εκλέγεται στην εκλογική περιφέρεια, αντίστοιχα.

– **N**

Γεγονός τύπου *form parliament* το οποίο σηματοδοτεί την εκλογή των μελών της Βουλής.

Αρχικά, θα πρέπει για κάθε κόμμα, να γίνει διάσχιση του δένδρου υποψήφιων του κόμματος και για όσους υποψηφίους έχουν εκλεγεί θα πρέπει να εισάγεται μια εγγραφή στη λίστα των βουλευτών, δηλαδή στη δομή του κοινοβουλίου (Parliament). Η δημιουργία της λίστας των βουλευτών θα πρέπει να γίνει ακολουθώντας ιδέες από τον αλγόριθμο συνένωσης δένδρων που είδατε στην **Άσκηση 2β της 3ης Σειράς Ασκήσεων**. Η λίστα του κοινοβουλίου θα πρέπει να είναι ταξινομημένη σε φθίνουσα σειρά ως προς το αναγνωριστικό <cid> του κάθε βουλευτή. **Η χρονική πολυπλοκότητα της συνένωσης θα πρέπει να είναι O(n), όπου n είναι ο συνολικός αριθμός των βουλευτών που εκλέγονται.**

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

```
N
  members
  <cid₁> <pid₁> <did₁>,
  <cid₂> <pid₂> <did₂>,
  …,
  <cidₙ> <pidₙ> <didₙ>
DONE
```

όπου n είναι ο αριθμός των εκλεγμένων βουλευτών και για κάθε i ∈ {1, …, n}, <cid$_i$>, <pid$_i$>, <did$_i$> είναι το αναγνωριστικό του i-οστού βουλευτή, το κόμμα στο οποίο ανήκει, και η περιφέρεια που εκπροσωπεί, αντίστοιχα.

– **I <did>**

Γεγονός τύπου *print district* το οποίο σηματοδοτεί την εκτύπωση στοιχείων από την εκλογική περιφέρεια με αναγνωριστικό <did>. Το γεγονός απαιτεί την αναζήτηση της εκλογικής περιφέρειας στον πίνακα εκλογικών περιφερειών. Κατά την εκτέλεση ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
I <did>
  seats <seats>
  blanks <blanks>
  invalids <invalids>
  partyVotes
  <pid₁> <votes₁>,
  <pid₂> <votes₂>,
  …,
  <pidₖ> <votesₖ>
DONE
```

όπου <seats> είναι ο αριθμός των εδρών που εκλέγει αυτή η εκλογική περιφέρεια, <blanks> είναι ο αριθμός των λευκών και <invalids> είναι ο αριθμός των άκυρων ψήφων στην εκλογική περιφέρεια. Επίσης k είναι ο αριθμός των κομμάτων που συμμετέχουν στις εκλογές και για κάθε i ∈ {1, …, k}, <pid$_i$> είναι το αναγνωριστικό του i -οστού κόμματος και <votes$_i$> ο αριθμός των ψήφων που έλαβε το κόμμα με αναγνωριστικό <pid$_i$> στην περιφέρεια με αναγνωριστικό <did>.

– **J <sid>**

Γεγονός τύπου *print station* το οποίο σηματοδοτεί την εκτύπωση στοιχείων από το εκλογικό τμήμα με αναγνωριστικό <sid>. Κατά την εκτέλεση ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
J <sid>
  registered n
  voters
  <vid₁> <voted₁>,
  <vid₂> <voted₂>,
  …,
  <vidₙ> <votedₙ>
DONE
```

όπου n είναι ο αριθμός των εγγεγραμμένων ψηφοφόρων στο εκλογικό τμήμα με αναγνωριστικό <sid> και για κάθε i ∈ {1, …, n}, $<vid_i>$ και $<voted_i>$ είναι το αναγνωριστικό του i-οστού ψηφοφόρου και το αν ο ψηφοφόρος έχει ασκήσει το εκλογικό του δικαίωμα, αντίστοιχα.

– **K <pid>**

Γεγονός τύπου *print party* το οποίο σηματοδοτεί την εκτύπωση στοιχείων από το κόμμα με αναγνωριστικό <pid>. Κατά την εκτέλεση ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
K <pid>
  elected
  <cid₁> <votes₁>,
  <cid₂> <votes₂>,
  …,
  <cidₙ> <votesₙ>
DONE
```

όπου n είναι ο αριθμός των εκλεγμένων υποψηφίων του κόμματος και για κάθε i ∈ {1, …, n}, $<cid_i>$ είναι το αναγνωριστικό του i-οστού υποψηφίου στο δένδρο των υποψήφιων του κόμματος, στη διάταξη που προκύπτει αν εφαρμοστεί ενδοδιατεταγμένη διάσχιση στο δένδρο και $<votes_i>$ είναι ο αριθμός των ψήφων που έλαβε ο υποψήφιος αυτός.

– **L**

Γεγονός τύπου *print parliament* το οποίο σηματοδοτεί την εκτύπωση των εκλεγμένων βουλευτών επικράτειας. Κατά την εκτέλεση ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
L
  members
  <cid1> <pid1> <did1>,
  <cid2> <pid2> <did2>,
  …,
  <cidn> <pidn> <didn>
DONE
```

όπου n είναι ο αριθμός των εκλεγμένων βουλευτών και για κάθε i ∈ {1, …, n}, <cid_i>, <pid_i>, <did_i> το αναγνωριστικό, το κόμμα, και η περιφέρεια που εκπροσωπεί ο i-οστός βουλευτής αντίστοιχα.

Bonus Section 🥳

- **BU <vid> <sid>** **[15M]**

  Γεγονός τύπου *bonus unregister voter* το οποίο σηματοδοτεί τη διαγραφή του ψηφοφόρου (Voter) με αναγνωριστικό <vid> από το εκλογικό τμήμα (Station) με αναγνωριστικό <sid>.

  Κατά το γεγονός αυτό εντοπίζεται η κατάλληλη αλυσίδα εκλογικών τμημάτων βάσει της συνάρτησης κατακερματισμού και στη συνέχεια εκτελείται αναζήτηση σε όλα τα δέντρα ψηφοφόρων μέχρι να βρεθεί και να διαγραφεί ο ζητούμενος ψηφοφόρος .

  Προσοχή: Η διαγραφή θα πρέπει να γίνεται με τρόπο ώστε το δένδρο να παραμένει πλήρες.

  Επιπλέον στο σημείο αυτό μειώνεται κατά ένα ο αριθμός εγγεγραμμένων ψηφοφόρων του συγκεκριμένου εκλογικού τμήματος (πεδίο registered του struct Station). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  ```
  BU <vid> <sid>
    Voters[<sid>]
    <vid₁>, <vid₂>, ..., <vidₙ>
  DONE
  ```

  όπου n είναι ο αριθμός των ψηφοφόρων του εκλογικού τμήματος με αναγνωριστικό <sid> και για κάθε i ∈ {1, …, n}, $<vid_i>$ είναι το αναγνωριστικό του i-οστού ψηφοφόρου στο δένδρο ψηφοφόρων του εκλογικού τμήματος <sid>, στη διάταξη που προκύπτει όταν εφαρμόζεται ενδοδιατεταγμένη διάσχιση δένδρο ψηφοφόρων του εκλογικού τμήματος με αναγνωριστικό <sid>

- **BF** **[5M]**

  Γεγονός τύπου *bonus free memory* το οποίο σηματοδοτεί την απελευθέρωση των κρατικών πόρων που χρησιμοποιήθηκαν για την πραγματοποίηση των εκλογών.

  Σε αυτό το γεγονός θα πρέπει να απελευθερώσετε τη μνήμη που δεσμεύσατε για την λειτουργία των δομών (και των περιεχόμενων δομών):

  - πίνακας εκλογικών περιφερειών (Districts)
  - πίνακας κατακερματισμού εκλογικών τμημάτων (StationsHT)
  - πίνακας κομμάτων (Parties)
  - κοινοβούλιο (Parliament)

  Όπως και να σιγουρευτείτε πως δεν έχετε διαρροές μνήμης (memory leaks) που οφείλονται σε άλλα σημεία του προγράμματός σας.

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

  ```
  BF
  DONE
  ```

Αρχικές τιμές πεδίων δομών δεδομένων

- Εγγραφή εκλογικής περιφέρειας **(**District):
  - **did**: λήψη από testfile. Για κενό κελί: -1.
  - **seats**: λήψη από testfile.
  - **blanks**: 0.
  - **invalids**: 0.
  - **partyVotes**: 0 σε κάθε κελί.
- Εγγραφή εκλογικού τμήματος (Station):
  - **sid:** λήψη από testfile.
  - **did:** λήψη από testfile.
  - **registered:** 0.
  - **voters:** NULL.
  - **next:** NULL.
- Εγγραφή ψηφοφόρου (Voter):
  - **sid:** λήψη από testfile.
  - **voted:** false.
  - **registered:** 0.
  - **parent:** NULL.
  - **lc:** NULL
  - **rc:** NULL.
- Εγγραφή κόμματος (Party):
  - **pid:** αριθμός αντίστοιχου κελιού του πίνακα κομμάτων (Parties)
  - **electedCount:** 0
  - **candidates:** NULL
- Εγγραφή υποψήφιου βουλευτή (Candidate):
  - **cid:** λήψη από testfile.
  - **did:** λήψη από testfile.
  - **votes:** 0.
  - **isElected:** false.
  - **lc:** NULL
  - **rc:** NULL
- **Κοινοβούλιο (Parliament)**: NULL.
- Εγγραφή εκλεγμένου βουλευτή (ElectedCandidate):
  - **cid:** αντιγραφή πεδίου από αντίστοιχο Candidate.
  - **did:** αντιγραφή πεδίου από αντίστοιχο Candidate.
  - **pid:** αντιγραφή πεδίου από αντίστοιχο Party.
  - **next:** NULL.

Βαθμολογία

A	announce elections	2
D	create district	3
S	create station	10
R	register voter	15
C	register candidate	15
V	vote	15
M	count votes	20
N	form parliament	15
I	print district	1
J	print station	2
K	print party	1
L	print parliament	1

Πληροφορίες Παράδοσης

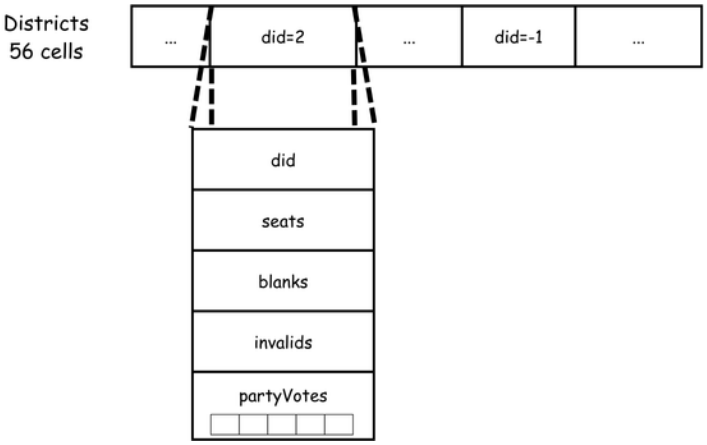
- Σε ένα ζευγάρι αρχείων πηγαίου κώδικα (πχ voting.c, voting.h) θα σας δοθούν οι ορισμοί των κατάλληλων τύπων για την υλοποίηση των ζητούμενων δομών, οι ορισμοί των κατάλληλων καθολικών μεταβλητών, οι υπογραφές των συναρτήσεων γεγονότων. Θα υλοποιήσετε την εργασία σας επεκτείνοντας τα αρχεία αυτά με κατάλληλο τρόπο.
- Κώδικας για parsing των γεγονότων από ένα input file θα σας δοθεί στο main source file (main.c). Η κλήση των γεγονότων πραγματοποιείται, επίσης, στο σώμα της main.
- Θα σας δοθεί ένα Makefile που χτίζει το εκτελέσιμο αρχείο από τα παραπάνω αρχεία πηγαίου κώδικα.
- Το παραδοτέο σας θα πρέπει να περιέχει μόνο τα παραπάνω αρχεία πηγαίου κώδικα και το Makefile. Δεν μπορείτε να προσθέσετε επιπλέον αρχεία.

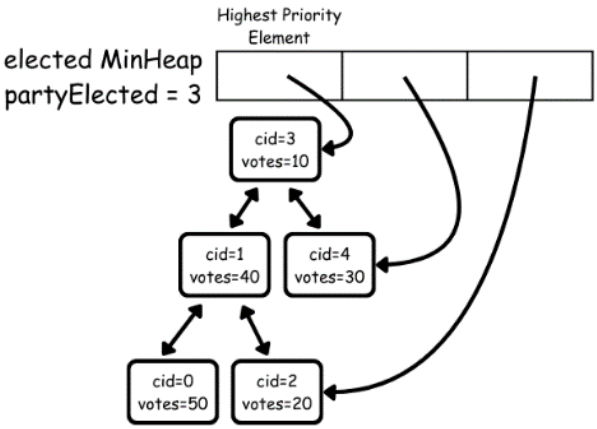
Μπορείτε να χτίσετε και στη συνέχεια να τρέξετε το εκτελέσιμο της εργασίας σας με τις ακόλουθες εντολές

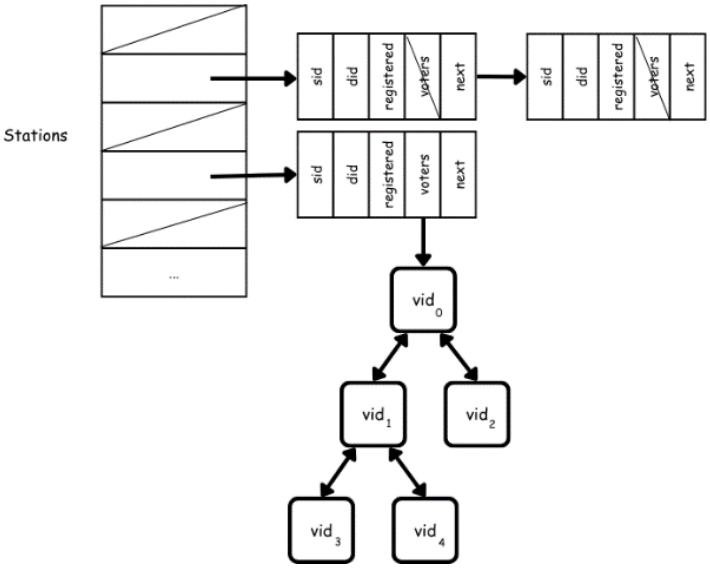
```
make
./main <input_file>
```

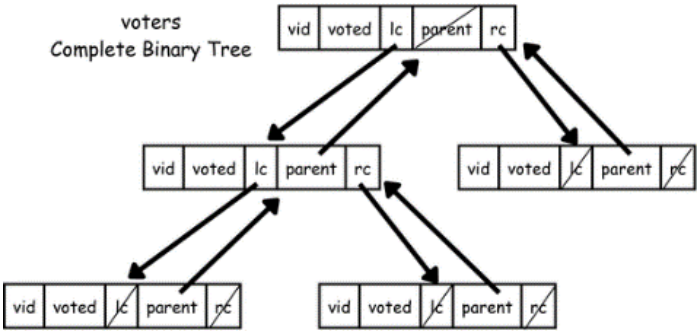
```
struct District {
    int did;
    int seats;
    int blanks;
    int invalids;
    int partyVotes[5];
};
struct Station {
    int sid;
    int did;
    int registered;
    struct Voter* voters;
    struct Station* next;
};
struct Voter {
    int vid;
    bool voted;
    Voter* parent;
    struct Voter* lc;
    struct Voter* rc;
};
struct Party {
    int pid;
    int electedCount;
    struct Candidate* candidates;
};
struct Candidate {
    int cid;
    int did;
    int votes;
    bool isElected;
    struct Candidate* lc;
    struct Candidate* rc;
};
```

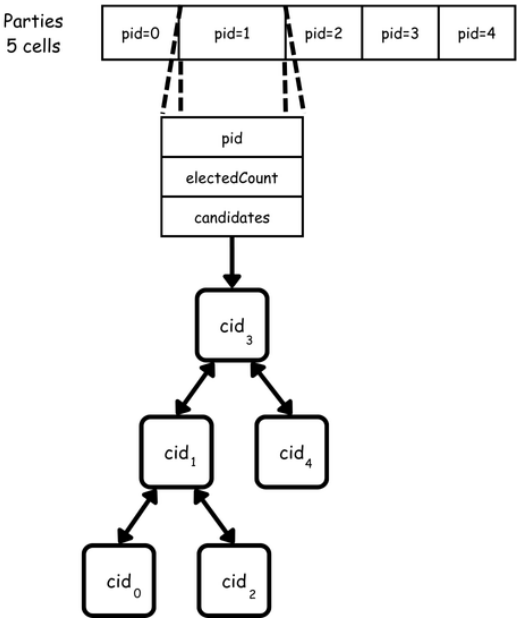
```
struct ElectedCandidate {
    int cid;
    int did;
    int pid;
    struct ElectedCandidate* next;
};
struct District Districts[56];
struct Station** StationsHT;
struct Party Parties[5];
struct ElectedCandidate* Parliament;
```

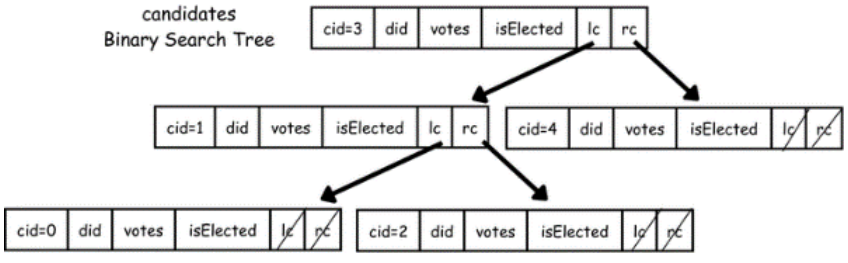

Districts
56 cells
...
did=2
...
did=-1
...
did
seats
blanks
invalids
partyVotes

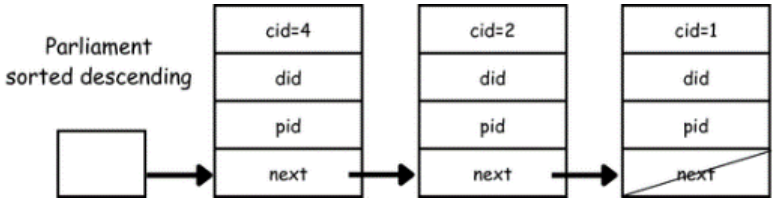

Highest Priority
Element
elected MinHeap
partyElected = 3
cid=3
votes=10
cid=1
votes=40
cid=4
votes=30
cid=0
votes=50
cid=2
votes=20


Stations
sid
did
registered
voters
next
vid0
vid1
vid2
vid3
vid4


voters
Complete Binary Tree
vid
voted
lc
parent
rc


Parties
5 cells
pid=0
pid=1
pid=2
pid=3
pid=4
pid
electedCount
candidates
cid3
cid1
cid4
cid0
cid2


candidates
Binary Search Tree
cid=3
did
votes
isElected
lc
rc
cid=1
cid=4
cid=0
cid=2


Parliament
sorted descending
cid=4
cid=2
cid=1
did
pid
next