HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο - Ακαδημαϊκό Έτος 2020-2021
Διδάσκουσα: Παναγιώτα Φατούρου
Προγραμματιστική Εργασία - 2ο Μέρος

Ημερομηνία Παράδοσης: Δευτέρα, 21 Δεκεμβρίου 2020, ώρα 09:59 πμ

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

Φωτογραφία: https://upload.wikimedia.org/wikipedia/en/7/72/AmongUs_CoverArt.jpg

Γενική Περιγραφή Εργασίας

Στην εργασία αυτή καλείστε να υλοποιήσετε μία προσομοίωση, μίας παρτίδας του παιχνιδιού Among Us.

«Among Us is an online multiplayer social deduction game, developed and published by American game studio InnerSloth and released on June 15, 2018. The game takes place in a space-themed setting where players each take on one of two roles, most being Crewmates, and a predetermined number being Impostors. The goal for the Crewmates is to identify the Impostors, remove them from the game, and complete tasks around the map; the Impostors' goal is to covertly sabotage the Crewmates and remove them from the game before they complete all their tasks. Through a plurality vote, players believed to be Impostors may be removed from the game. If all Impostors are removed from the game or all tasks are completed, the Crewmates win; if there are an equal number of Impostors and Crewmates, or if a critical sabotage goes unresolved, the Impostors win.» (Wikipedia)

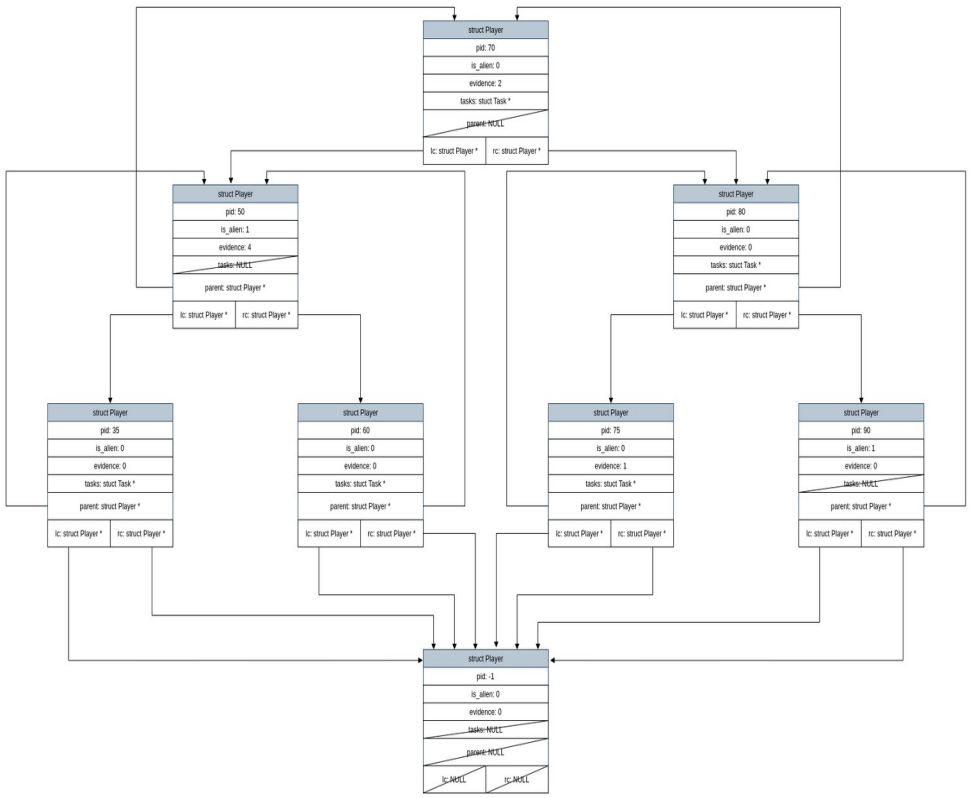
Αξίζει να σημειωθεί, ότι για τους εκπαιδευτικούς σκοπούς του μαθήματος ορισμένα σημεία της εργασίας ενδέχεται να αποκλίνουν από το πραγματικό παιχνίδι.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Κάθε παίκτης (player), εξωγήινος (impostor) ή άνθρωπος (crewmate), περιγράφεται από ένα μοναδικό αναγνωριστικό τύπου int (pid). Πληροφορίες για τους παίκτες αποθηκεύονται σε ένα **διπλά συνδεδεμένο δυαδικό δένδρο αναζήτησης με κόμβο φρουρό, το οποίο ονομάζεται δένδρο παικτών (Σχήμα 1).** Κάθε κόμβος του δένδρου είναι ένα struct τύπου Player και αντιστοιχεί σε έναν παίκτη. Για κάθε παίκτη, υπάρχει μία μεταβλητή τύπου int (που λέγεται is_alien), η οποία υποδηλώνει αν ο παίκτης είναι εξωγήινος ή άνθρωπος και μία μεταβλητή τύπου int (που λέγεται evidence), η οποία δείχνει κατά πόσο ο παίκτης θα θεωρηθεί ύποπτος σε επόμενες ψηφοφορίες. Σημειώνεται ότι, παρότι γίνεται καταγραφή του τύπου του κάθε παίκτη στο δένδρο, θεωρούμε ότι οι άλλοι παίκτες δεν έχουν πρόσβαση στο πεδίο is_allien κάθε παίκτη και άρα δεν μπορούν να ανακαλύψουν από εκεί, αν ο παίκτης είναι εξωγήινος ή άνθρωπος. Το δένδρο παικτών είναι ταξινομημένο (σύμφωνα με την ενδοδιατεταγμένη διάσχιση) ως προς το πεδίο pid των παικτών που περιέχει.

Πιο συγκεκριμένα, μια εγγραφή τύπου struct Player έχει τα ακόλουθα πεδία:

- **pid**: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον παίκτη.
- **is_alien**: Μεταβλητή (τύπου int) που σηματοδοτεί το αν ο παίκτης είναι εξωγήινος (impostor) ή άνθρωπος (crewmate).
- **evidence**: Ακέραιος που περιγράφει το βαθμό κατά τον οποίο ένας παίκτης θεωρείται ύποπτος (τύπου int).
- **parent:** Δείκτης (τύπου struct Player) που δεικτοδοτεί το **γονικό κόμβο** του κόμβου με αναγνωριστικό pid.
- **lc:** Δείκτης (τύπου struct Player) που δεικτοδοτεί τον **αριστερό θυγατρικό κόμβο** του κόμβου με αναγνωριστικό pid.
- **rc:** Δείκτης (τύπου struct Player) που δεικτοδοτεί το **δεξιό θυγατρικό κόμβο** του κόμβου με αναγνωριστικό pid.
- **tasks**: Δείκτης (τύπου *struct Task*) στη ρίζα ενός απλά-συνδεδεμένου δυαδικού δένδρου αναζήτησης (χωρίς κόμβο φρουρό), το οποίο περιέχει πληροφορίες για τις εργασίες (tasks) που θα ανατεθούν στον παίκτη.



Σχήμα 1. Το δένδρο παικτών που είναι διπλά συνδεδεμένο, με κόμβο φρουρό και ταξινομημένο. Το δεύτερο πεδίο δείχνει αν ένας παίκτης είναι εξωγήινος ή όχι και το τρίτο πεδίο φανερώνει κατά πόσο ύποπτος είναι ένας παίκτης.

Ο κάθε παίκτης έχει διάφορες εργασίες (tasks) να ολοκληρώσει. Έτσι, για κάθε παίκτη υπάρχει ένα **δυαδικό δένδρο αναζήτησης, το οποίο ονομάζεται δένδρο εργασιών του παίκτη.** Κάθε εργασία που έχει ανατεθεί στον παίκτη χαρακτηρίζεται από ένα μοναδικό αναγνωριστικό τύπου int (tid) και έχει ένα βαθμό δυσκολίας (που αναγράφεται στο πεδίο difficulty, επίσης τύπου int). Επιπρόσθετα, κάθε κόμβος του δένδρου εργασιών ενός παίκτη έχει ένα ακόμη πεδίο lcnt που αποθηκεύει έναν ακέραιο. Το δένδρο εξασφαλίζει την εξής ιδιότητα που ονομάζουμε **ιδιότητα μετρητών του δένδρου**: *«Ο ακέραιος lcnt κάθε κόμβου ν του δένδρου αποθηκεύει τον αριθμό των κόμβων που περιέχονται στο αριστερό υποδένρο του ν.»* Το δένδρο εργασιών του κάθε παίκτη είναι ταξινομημένο ως προς το tid των εργασιών που περιέχει.

Κάθε κόμβος του δένδρου εργασιών ενός συγκεκριμένου παίκτη, αποτελεί μία εγγραφή τύπου struct Task με τα ακόλουθα πεδία:

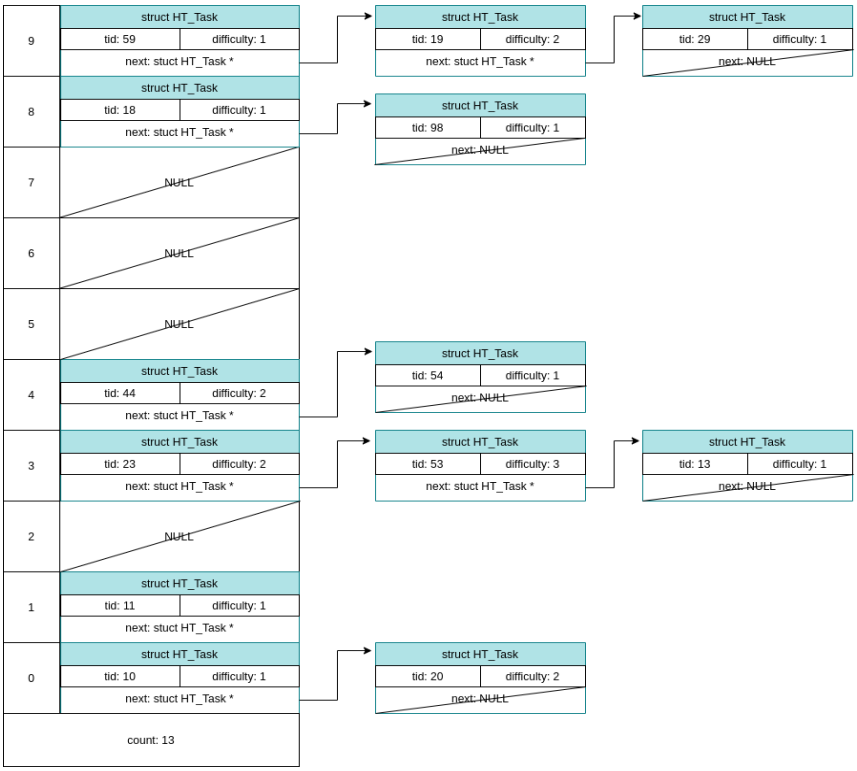
- **tid**: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά την κάθε εργασία.
- **difficulty**: Ο βαθμός δυσκολίας (τύπου int) της κάθε εργασίας. Υπάρχουν 3 βαθμοί δυσκολίας (που σηματοδοτούνται από τους αριθμούς 1, 2 και 3).
- **lcnt**: Το πλήθος των κόμβων του αριστερού υποδένδρου (τύπου int) του κόμβου με αναγνωριστικό tid.
- **lc:** Δείκτης (τύπου struct Task) που δεικτοδοτεί τον **αριστερό θυγατρικό κόμβο** του κόμβου με αναγνωριστικό tid.
- **rc:** Δείκτης (τύπου struct Task) που δεικτοδοτεί τον **δεξιό θυγατρικό κόμβο** του κόμβου με αναγνωριστικό tid.

Όλες οι εργασίες που πρέπει να πραγματοποιηθούν για να νικήσει το πλήρωμα του διαστημοπλοίου (άνθρωποι), εισάγονται αρχικά σε έναν **πίνακα κατακερματισμού, που ονομάζεται πίνακας κατακερματισμού γενικών εργασιών** (**Σχήμα 2**). **Η επίλυση των συγκρούσεων γίνεται βάσει της μεθόδου των ξεχωριστών αλυσίδων. Η συνάρτηση κατακερματισμού επιλέγεται βάσει της τεχνικής του καθολικού κατακερματισμού.** Για την υλοποίηση του καθολικού κατακερματισμού, παρέχεται ένας πίνακας primes_g[650] με πρώτους αριθμούς σε αύξουσα σειρά, το μέγιστο πλήθος εργασιών μέσω της μεταβλητής max_tasks_g και το μέγιστο αναγνωριστικό μεταξύ των εργασιών, μέσω της μεταβλητής max_tid_g. Κάθε κόμβος οποιασδήποτε αλυσίδας του πίνακα κατακερματισμού γενικών εργασιών αποτελεί μία εγγραφή τύπου struct HT_Task που περιέχει τα ακόλουθα πεδία:

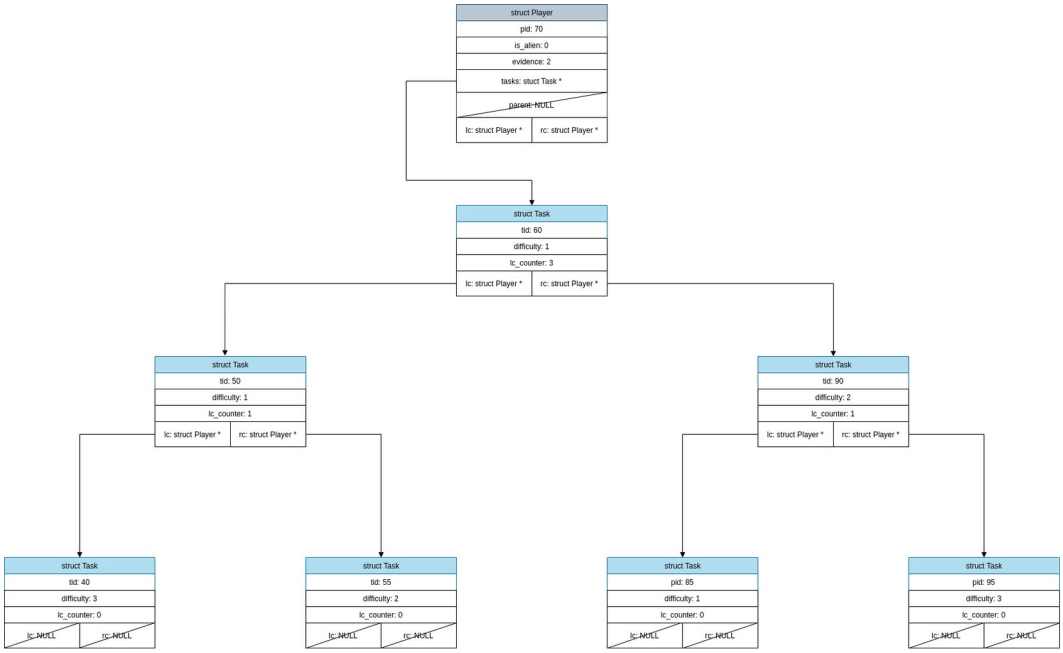
- **tid**: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά την κάθε εργασία.
- **difficulty**: Ο βαθμός δυσκολίας (τύπου int) της κάθε εργασίας. Υπάρχουν 3 βαθμοί δυσκολίας (που σηματοδοτούνται από τους αριθμούς 1, 2 και 3).
- **next**: Δείκτης (τύπου struct HT_Task) στον επόμενο κόμβο της αλυσίδας.

Υπάρχει ένα struct, τύπου General_Tasks_HT που περιέχει τα εξής πεδία:

- **count**: που αποθηκεύει το συνολικό πλήθος των εργασιών που αποθηκεύονται στο πίνακα εργασιών.
- **tasks**: Δείκτης σε struct τύπου HT_Task που υλοποιεί το γενικό πίνακα εργασιών (ο οποίος δεσμεύεται δυναμικά).



Σχήμα 2. Ο πίνακας κατακερματισμού γενικών εργασιών. Οι εργασίες εισάγονται στην αντίστοιχη αλυσίδα βάσει του αναγνωριστικού τους. Στο παράδειγμα, η συνάρτηση κατακερματισμού είναι η h(tid) = tid mod 10.



Σχήμα 3. Το ταξινομημένο δένδρο εργασιών ενός παίκτη.

Αφότου όλες οι εργασίες εισαχθούν στο γενικό πίνακα κατακερματισμού εργασιών, θα μοιραστούν τελικά στα δένδρα εργασιών των μελών του πληρώματος (**Σχήμα 3**). Στη συνέχεια, εκείνοι αρχίζουν να τις υλοποιούν. Κάθε φορά που κάποιος άνθρωπος ολοκληρώνει μία εργασία, τότε αυτή αφαιρείται από το δένδρο των εργασιών του και εισάγεται σε μία **ουρά προτεραιότητας μεγίστων, που λέγεται ουρά ολοκληρωμένων εργασιών** (**Σχήμα 4**). Κάθε στοιχείο της ουράς ολοκληρωμένων εργασιών αποτελεί μία εγγραφή τύπου struct HT_Task (δηλαδή είναι ιδίου τύπου με τους κόμβους των αλυσίδων του πίνακα κατακερματισμού γενικών εργασιών των παικτών, με τη διαφορά ότι το πεδίο next δεν χρησιμοποιείται). Η προτεραιότητα κάθε στοιχείου της ουράς ορίζεται από την τιμή του πεδίου difficulty (μεγαλύτερες τιμές σηματοδοτούν μεγαλύτερη προτεραιότητα).

Η ουρά ολοκληρωμένων εργασιών υλοποιείται με ένα struct τύπου Completed_Tasks_PQ που περιέχει τα εξής πεδία:

- s**ize**: Ακέραιος που αποθηκεύει το πλήθος των στοιχείων της ουράς προτεραιότητας την εκάστοτε χρονική στιγμή.
- **tasks[]**: Ένας πίνακας (με στοιχεία τύπου struct HT_Task) που θα δεσμεύεται δυναμικά και θα περιέχει τα στοιχεία της ουράς προτεραιότητας,

	0	1	2	3	4	5	6	7	8
size: 8	struct HT_Task	struct HT_Task	struct HT_Task	struct HT_Task	struct HT_Task	struct HT_Task	struct HT_Task	struct HT_Task	struct HT_Task
	tid: 13	tid: 11	tid: 29	tid: 70	tid: 10	tid: 8	tid: 60	tid: 34	tid: 40
	difficulty: 3	difficulty: 2	difficulty: 2	difficulty: 1	difficulty: 1	difficulty: 1	difficulty: 1	difficulty: 1	difficulty: 1
	next: NULL	next: NULL	next: NULL	next: NULL	next: NULL	next: NULL	next: NULL	next: NULL	next: NULL

Σχήμα 4. Η ουρά προτεραιότητας μεγίστων με τις ολοκληρωμένες εργασίες. Αν το πλήθος των στοιχείων της είναι ίσο με το πλήθος των στοιχείων του πίνακα κατακερματισμού γενικών εργασιών τότε οι άνθρωποι νίκησαν την παρτίδα.

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-fle> είναι το όνομα ενός αρχείου εισόδου (π.χ. testile) το οποίο περιέχει γεγονότα των ακόλουθων μορφών:

P <pid> <is_alien>

Γεγονός τύπου insert Player που υποδηλώνει την εισαγωγή ενός παίκτη με αναγνωριστικό <pid> στο παιχνίδι. Κατά το γεγονός αυτό ένα στοιχείο (που θα αντιστοιχεί στο νέο παίκτη) θα εισάγεται στο δένδρο παικτών. Τα πεδία evidence και is_alien του στοιχείου αυτού πρέπει να αρχικοποιούνται με τις τιμές 0 και <is_allien>, αντίστοιχα (αν η <is_allien> είναι 1 τότε ο παίκτης είναι εξωγήινος, διαφορετικά είναι άνθρωπος, μέλος του πληρώματος). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

P <pid> <is_alien>
Players = <pid_1:is_alien$_1$><pid_2:is_alien$_2$> … <pid_n:is_alien$_n$>
DONE

όπου n είναι ο αριθμός των κόμβων στο δένδρο παικτών και για κάθε i ϵ{1,...,n}, <pid_i> είναι το αναγνωριστικό του παίκτη που αντιστοιχεί στον i-οστό κόμβο του δένδρου σύμφωνα με την ενδοδιατεταγμένη διάσχιση.

T <tid> <difficulty>

Γεγονός τύπου Insert Task που υποδηλώνει τη δημιουργία μιας εργασίας με αναγνωριστικό <tid> και βαθμό δυσκολίας <difficulty>. Κατά το γεγονός αυτό, ένα νέο στοιχείο, που αντιστοιχεί σ' αυτήν την εργασία, θα εισάγεται στον πίνακα κατακερματισμού γενικών εργασιών. Συγκεκριμένα, το αναγνωριστικό <tid> θα πρέπει να δίνεται ως είσοδος στην συνάρτηση κατακερματισμού και στη συνέχεια ο νέας κόμβος θα εισάγεται στην αντίστοιχη αλυσίδα του πίνακα. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

T <tid> <difficulty>
Chain 0: <$tid_{0,1}$, $difficulty_{0,1}$>, <$pid_{0,2}$, $difficulty_{0,2}$>, ..., <$aid_{0,N0}$, $difficulty_{0,N0}$>
Chain 1: <$tid_{1,1}$, $difficulty_{1,1}$ >, <$aid_{1,2}$, $difficulty_{1,2}$>, ..., <$aid_{1,N1}$, $difficulty_{0,N1}$>
...
Chain n: <$tid_{n,1}$, $difficulty_{n,1}$>, <$aid_{n,2}$, $difficulty_{n,2}$>, ..., <$aid_{n,Nn}$, $difficulty_{n,Nn}$>
DONE

όπου n είναι το πλήθος των θέσεων του πίνακα κατακερματισμού γενικών εργασιών και για κάθε i, $0 \le i \le n-1$, N_i είναι το πλήθος των κόμβων στην αλυσίδα της θέσης i του πίνακα κατακερματισμού και για κάθε j, $1 \le j \le N_i$, $tid_{j,i}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της εργασίας που αντιστοιχεί στον j-οστό κόμβο της αλυσίδας που βρίσκεται στην θέση i του πίνακα κατακερματισμού και η δυσκολία εκτέλεσής της, αντίστοιχα.

D

Γεγονός τύπου Distribute Tasks που υποδηλώνει την ανάθεση εργασιών στους παίκτες. Κατά το γεγονός αυτό, θα πρέπει να μοιράζονται οι εργασίες που βρίσκονται στο πίνακα κατακερματισμού γενικών εργασιών στους παίκτες ακολουθώντας έναν αλγόριθμο εκ περιτροπής (round robin). Συγκεκριμένα, πραγματοποιείται διάσχιση στον πίνακα κατακερματισμού από την πρώτη προς την τελευταία αλυσίδα του και η i-οστή εργασία που διασχίζεται ανατίθεται στον i-οστό παίκτη (MOD n) σύμφωνα με την ενδοδιατεταγμένη διάσχιση, όπου n είναι το πλήθος των στοιχείων του δένδρου παικτών. Για τη διάσχιση του πίνακα κατακερματισμού απαιτείται η χρήση μετρητών.

Για την υλοποίηση αυτού του γεγονότος θα πρέπει να πραγματοποιείται μια αναδρομική διάσχιση του δένδρου και σε κάθε κόμβο που δεν είναι εξωγήινος θα πρέπει να ανατίθεται από μια εργασία. Η παραπάνω διαδικασία εκτελείται επαναληπτικά μέχρι να ανατεθούν όλες οι εργασίες στους παίκτες. Επομένως, θα πρέπει να εκτελούνται μια σειρά από διαδοχικές διασχίσεις του δένδρου και σε κάθε τέτοια διάσχιση θα πρέπει να ανατίθεται μία εργασία σε καθέναν από τους παίκτες-μέλη του πληρώματος. Η χρονική πολυπλοκότητα αυτού του γεγονότος θα πρέπει να είναι O(n), όπου n είναι ο συνολικός αριθμός εργασιών στο γενικό πίνακα εργασιών.

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

D

$Player_1$ = <$tid_{1,1}$,$difficulty_{1,1}$>,<$tid_{1,2}$,$difficulty_{1,2}$> … <$tid_{1,mn}$,$difficulty_{1,m1}$>

$Player_2$ = <$tid_{2,1}$,$difficulty_{2,1}$>,<$tid_{2,2}$,$difficulty_{2,2}$> … <$tid_{2,mn}$,$difficulty_{2,m2}$>

…

$Player_n$ = <$tid_{n,1}$,$difficulty_{n,1}$>,<$tid_{n,2}$,$difficulty_{n,2}$>… <$tid_{n,mn}$,$difficulty_{n,mn}$>

DONE

όπου n είναι το πλήθος των παικτών στο δένδρο παικτών και για κάθε i, $1 \le i \le n$, m_i είναι το πλήθος των κόμβων του δένδρου εργασιών του i-οστού παίκτη στο δένδρο παικτών (σύμφωνα με την ενδοδιατεταγμένη διάσχιση), και για κάθε $j \in \{1,...,m_i\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της j-οστής εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, στο δένδρο εργασιών του i-οστού παίκτη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).

I <pid> <tid>

Γεγονός τύπου Implement Task, που υποδηλώνει ότι ο παίκτης με το αναγνωριστικό <pid>, υλοποιεί μία εργασία με αναγνωριστικό <tid>. Κατά το γεγονός αυτό θα πρέπει να πραγματοποιούνται οι ακόλουθες ενέργειες. Αρχικά, θα αναζητάτε τον συγκεκριμένο παίκτη στο δένδρο παικτών. Εφόσον έχετε βρει τον συγκεκριμένο παίκτη, θα αναζητάτε την εργασία t με αναγνωριστικό <tid> στό δένδρο εργασιών του παίκτη. Στη συνέχεια εφόσον βρείτε την t, θα τη διαγράφετε από το δένδρο εργασιών του παίκτη (έτσι ώστε το δένδρο να παραμείνει ταξινομημένο και οι μετρητές αριστερών κόμβων να έχουν ενημερωθεί σωστά μετά το πέρας του γεγονότος). Τέλος, θα εισάγεται έναν κόμβο για την t στην ουρά ολοκληρωμένων εργασιών. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

I <pid> <tid>

$Player_1$ = <$tid_{1,1}$,$difficulty_{1,1}$>,<$tid_{1,2}$,$difficulty_{1,2}$> … <$tid_{1,mn}$,$difficulty_{1,mn}$>

$Player_2$ = <$tid_{2,1}$,$difficulty_{2,1}$>,<$tid_{2,2}$,$difficulty_{2,2}$> … <$tid_{2,mn}$,$difficulty_{2,mn}$>

…

$Player_n$ = <$tid_{n,1}$,$difficulty_{n,1}$>,<$tid_{n,2}$,$difficulty_{n,2}$>… <$tid_{mn,mn}$,$difficulty_{mn,mn}$>

DONE

όπου n είναι το πλήθος των παικτών στο δένδρο παικτών και για κάθε i, $1 \leq i \leq n$, m_i είναι το πλήθος των κόμβων του δένδρου εργασιών του i-οστού παίκτη στο δένδρο παικτών (σύμφωνα με την ενδοδιατεταγμένη διάσχιση), και για κάθε $j \in \{1,...,m_i\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της j-οστής εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, στο δένδρο εργασιών του i-οστού παίκτη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).

E <pid_1> <pid_2>

Γεγονός τύπου Eject Player που υποδηλώνει το άγγιγμα του παίκτη με αναγνωριστικό <pid_1> από ένα εξωγήινο και την αφαίρεσή του από το διαστημόπλοιο (χωρίς να το αντιληφθεί οποιοσδήποτε άλλος παίκτης). Κατά το γεγονός αυτό, ο παίκτης με αναγνωριστικό <pid_1> αφαιρείται από το δένδρο παικτών και το δένδρο εργασιών του παίκτη συγχωνεύεται με εκείνο του παίκτη με αναγνωριστικό <pid_2>. Το δένδρο εργασιών του παίκτη με αναγνωριστικό <pid_2> που θα προκύψει από τη συγχώνευση πρέπει να είναι ταξινομημένο, να έχει ύψος O(logn) και οι μετρητές αριστερών κόμβων να έχουν σωστές τιμές μετά την συγχώνευση. Η διαδικασία της συγχώνευσης πρέπει να πραγματοποιείται σε χρόνο O(n1+n2), όπου n1 και n2 είναι το πλήθος των κόμβων στα δένδρα εργασιών των παικτών <pid1> και <pid2>, αντίστοιχα. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

E <pid_1> <pid_2>

$Player_1$ = <$tid_{1,1}$,$difficulty_{1,1}$>,<$tid_{1,2}$,$difficulty_{1,2}$> … <$tid_{1,mn}$,$difficulty_{1,mn}$>

$Player_2$ = <$tid_{2,1}$,$difficulty_{2,1}$>,<$tid_{2,2}$,$difficulty_{2,2}$> … <$tid_{2,mn}$,$difficulty_{2,mn}$>

…

$Player_n$ = <$tid_{n,1}$,$difficulty_{n,1}$>,<$tid_{n,2}$,$difficulty_{n,2}$>… <$tid_{mn,mn}$,$difficulty_{mn,mn}$>

DONE

όπου n είναι το πλήθος των παικτών στο δένδρο παικτών και για κάθε i, $1 \leq i \leq n$, m_i είναι το πλήθος των κόμβων του δένδρου εργασιών του i-οστού παίκτη στο δένδρο παικτών (σύμφωνα με την ενδοδιατεταγμένη διάσχιση), και για κάθε $j \in \{1,...,m_i\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της j-οστής εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, στο δένδρο εργασιών του i-οστού παίκτη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).

W <pid_1> <pid_2> <pid_a> <number_witnesses>

Γεγονός τύπου Witness Ejection που υποδηλώνει το άγγιγμα του παίκτη με αναγνωριστικό <pid_1> από τον εξωγήινο με αναγνωριστικό <pid_a> και την αφαίρεσή του παίκτη <pid_1> από το διαστημόπλοιο. Ωστόσο, στο γεγονός αυτό, την αφαίρεση του παίκτη <pid_1> την βλέπουν <number_witnesses> άνθρωποι (crewmates). Το γεγονός αυτό εκτελείται με τον ίδιο ακριβώς τρόπο όπως το event E: ο παίκτης με αναγνωριστικό <pid_1> αφαιρείται από το δένδρο παικτών και το δένδρο εργασιών του συγχωνεύεται με εκείνο του παίκτη με αναγνωριστικό <pid_2>. Ωστόσο, θα πρέπει να γίνει επιπρόσθετα το εξής: να βρείτε τον εξωγήινο με αναγνωριστικό <pid_a> και να ενημερώσετε το πεδίο evidence για αυτόν προσθέτοντάς του τον αριθμό <number_witnesses>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

W <pid$_1$> <pid$_2$> <pid$_a$> <number_witnesses>

<Player$_1$, evidence$_1$> = <tid$_{1,1}$,difficulty$_{1,1}$>,<tid$_{1,2}$,difficulty$_{1,2}$> … <tid$_{1,mn}$,difficulty$_{1,mn}$>

<Player$_2$, evidence$_2$> = <tid$_{2,1}$,difficulty$_{2,1}$>,<tid$_{2,2}$,difficulty$_{2,2}$> … <tid$_{2,mn}$,difficulty$_{2,mn}$>

…

<Player$_n$, evidence$_n$> = <tid$_{n,1}$,difficulty$_{n,1}$>,<tid$_{n,2}$,difficulty$_{n,2}$>… <tid$_{mn,mn}$,difficulty$_{mn,mn}$>

DONE

όπου n είναι το πλήθος των παικτών στο δένδρο παικτών και για κάθε i, $1 \leq i \leq n$, m_i είναι το πλήθος των κόμβων του δένδρου εργασιών του i-οστού παίκτη στο δένδρο παικτών (σύμφωνα με την ενδοδιατεταγμένη διάσχιση), και για κάθε $j \in \{1,...,m_i\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της j-οστής εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, στο δένδρο εργασιών του i-οστού παίκτη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).

S <number_of_tasks> <pid>

Γεγονός τύπου Sabotage, το οποίο υποδηλώνει ότι ένας εξωγήινος αφαιρεί <number_of_tasks> εργασίες από την ουρά προτεραιότητας ολοκληρωμένων εργασιών και τις ξαναμοιράζει στους παίκτες. Κατά το γεγονός αυτό, πραγματοποιούνται <number_of_tasks> διαγραφές εκείνων των εργασιών με το μεγαλύτερο βαθμό δυσκολίας (δηλαδή με τη μεγαλύτερη προτεραιότητα) από την ουρά ολοκληρωμένων εργασιών. Κάθε μια εργασία που αφαιρείται από την ουρά προτεραιότητας, ανατίθεται σε κάποιον παίκτη ακολουθώντας τον εξής αλγόριθμο:

- Αρχικά, βρίσκουμε τον παίκτη με αναγνωριστικό <pid> στο δένδρο παικτών.
- Η πρώτη εργασία ανατίθεται στον παίκτη p που προηγείται του παίκτη <pid> κατά $\lfloor$number_of_tasks/2$\rfloor$ θέσεις στην ενδοδιατεταγμένη διάσχιση (ή, αν ο p είναι εξωγήινος, στον πρώτο κόμβο που αντιστοιχεί σε άνθρωπο από τους προηγούμενους του p). Για να πραγματοποιηθεί αυτό, χρειάζεται να υλοποιήσετε τη συνάρτηση FindInorderPredecessor(), η οποία βρίσκει τον προηγούμενο ενός κόμβου στην ενδοδιατεταγμένη διάσχιση. Η διαδικασία αυτή θα πρέπει να κληθεί επαναληπτικά μέχρι να προσεγγίσετε το συγκεκριμένο παίκτη.
- Κάθε επόμενη εργασία ανατίθεται σε καθέναν από τους επόμενους κόμβους του p σύμφωνα με την ενδοδιατεταγμένη διάσχιση, που δεν είναι εξωγήινοι. Για να πραγματοποιηθεί αυτό, απαιτείται να υλοποιήσετε τη συνάρτηση, FindInorderSuccessor(), η οποία βρίσκει τον επόμενο ενός κόμβου στην ενδοδιατεταγμένη διάσχιση.
- Με τον τρόπο αυτό, θα ανατεθεί από μια εργασία σε καθέναν από τους $\lfloor$number_of_tasks/2$\rfloor$ προηγούμενους κόμβους του p που αντιστοιχούν σε μέλη του πληρώματος (σύμφωνα με την ενδοδιατεταγμένη διάσχιση) και στους $\lfloor$number_of_tasks/2$\rfloor$ επόμενους κόμβους του p (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).
- Θεωρήστε ότι ο προηγούμενος του πρώτου κόμβου στην ενδοδιατεταγμένη διάσχιση είναι ο τελευταίος κόμβος (στην ενδοδιατεταγμένη διάσχιση). Επίσης, θεωρήστε ότι ο επόμενος του τελευταίου κόμβου (στην ενδοδιατεταγμένη διάσχιση) είναι ο πρώτος κόμβος (στην ενδοδιατεταγμένη διάσχιση).

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

S <number_of_tasks> <pid>

$Player_1$ = <$tid_{1,1}$,$difficulty_{1,1}$>,<$tid_{1,2}$,$difficulty_{1,2}$> … <$tid_{1,mn}$,$difficulty_{1,mn}$>

$Player_2$ = <$tid_{2,1}$,$difficulty_{2,1}$>,<$tid_{2,2}$,$difficulty_{2,2}$> … <$tid_{2,mn}$,$difficulty_{2,mn}$>

…

$Player_n$ = <$tid_{n,1}$,$difficulty_{n,1}$>,<$tid_{n,2}$,$difficulty_{n,2}$>… <$tid_{mn,mn}$,$difficulty_{mn,mn}$>

DONE

όπου n είναι το πλήθος των παικτών στο δένδρο παικτών και για κάθε i, $1 \le i \le n$, m_i είναι το πλήθος των κόμβων του δένδρου εργασιών του i-οστού παίκτη στο δένδρο παικτών (σύμφωνα με την ενδοδιατεταγμένη διάσχιση), και για κάθε $j \in \{1,...,m_i\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της j-οστής εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, στο δένδρο εργασιών του i-οστού παίκτη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).

V <pid_1> <pid_2> <vote_evidence>

Γεγονός τύπου Voting, το οποίο υποδηλώνει την ψηφοφορία και την απομάκρυνση ενός παίκτη (impostor ή crewmate) από το διαστημόπλοιο. Κατά το γεγονός αυτό, συμβαίνουν τα εξής. Αρχικά, ο παίκτης με αναγνωριστικό <pid_1> θεωρείται ύποπτος (γιατί θεωρούμε πως είναι αυτός που κάνει την περισσότερη φασαρία κατά τη ψηφοφορία) και έτσι το πεδίο evidence του struct που του αναλογεί στο δένδρο παικτών αυξάνει κατά <vote_evidence>. Στη συνέχεια, πρέπει να διατρέξετε το δένδρο παικτών, να βρείτε τον παίκτη με το μεγαλύτερο evidence και να τον αφαιρέσετε. Το δένδρο εργασιών του παίκτη που αποβάλλεται από το διαστημόπλοιο θα συγχωνευτεί με εκείνο του παίκτη με αναγνωριστικό <pid_2> (όπως έχει συζητηθεί στα γεγονότα E και W). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

V <pid_1> <pid_2> <vote_evidence>

<$Player_1$, $evidence_1$> = <$tid_{1,1}$,$difficulty_{1,1}$>,<$tid_{1,2}$,$difficulty_{1,2}$> … <$tid_{1,mn}$,$difficulty_{1,mn}$>

<$Player_2$, $evidence_2$> = <$tid_{2,1}$,$difficulty_{2,1}$>,<$tid_{2,2}$,$difficulty_{2,2}$> … <$tid_{2,mn}$,$difficulty_{2,mn}$>

…

<$Player_n$, $evidence_n$> = <$tid_{n,1}$,$difficulty_{n,1}$>,<$tid_{n,2}$,$difficulty_{n,2}$>… <$tid_{mn,mn}$,$difficulty_{mn,mn}$>

DONE

όπου n είναι το πλήθος των παικτών στο δένδρο παικτών και για κάθε i, $1 \le i \le n$, $Player_i$ και $evidence_i$ είναι ο i-οστός παίκτης και το evidence που του αναλογεί (αντίστοιχα) στο δένδρο παικτών (σύμφωνα με την ενδοδιατεταγμένη διάσχιση), m_i είναι το πλήθος των κόμβων του δένδρου εργασιών του i-οστού παίκτη στο δένδρο παικτών, και για κάθε $j \in \{1,...,m_i\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της j-οστής εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, στο δένδρο εργασιών του i-οστού παίκτη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).

G <pid_1> <pid_2>

Γεγονός τύπου Give Away Work, το οποίο υποδηλώνει τη μεταφορά εργασιών από το παίκτη με αναγνωριστικό <pid_1> σε ένα καινούριο παίκτη με αναγνωριστικό <pid_2>. Κατά το γεγονός αυτό θα εισάγεται τον παίκτη με αναγνωριστικό <pid_2> στο δένδρο παικτών. Έπειτα θα αναζητήσετε τον παίκτη με αναγνωριστικό <pid_1> στο δένδρο παικτών και θα μεταφέρετε τις μισές εργασίες που ανήκουν στον παίκτη <pid_1>, στον παίκτη με αναγνωριστικό <pid_2>. Για να πραγματοποιηθεί αυτό το γεγονός, θα πρέπει να υλοποιήσετε έναν αλγόριθμο που θα μετράει ποιος είναι ο συνολικός αριθμός κόμβων του δένδρου εργασιών T του παίκτη <pid1>. Στη συνέχεια, θα πρέπει να διασπάτε το T σε δύο άλλα δένδρα, με το καθένα να περιέχει τα μισά στοιχεία από ότι το T. Ο αλγόριθμός σας θα πρέπει να εκτελείται σε χρόνο O(h), όπου h είναι το ύψος του δένδρου των εργασιών του παίκτη <pid_1>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

G <pid_1> <pid_2>
Player$_1$ = <$tid_{1,1}$,difficulty$_{1,1}$>,<$tid_{1,2}$,difficulty$_{1,2}$> … <$tid_{1,mn}$,difficulty$_{1,mn}$>
Player$_2$ = <$tid_{2,1}$,difficulty$_{2,1}$>,<$tid_{2,2}$,difficulty$_{2,2}$> … <$tid_{2,mn}$,difficulty$_{2,mn}$>
…
Player$_n$ = <$tid_{n,1}$,difficulty$_{n,1}$>,<$tid_{n,2}$,difficulty$_{n,2}$>… <$tid_{mn,mn}$,difficulty$_{mn,mn}$>
DONE

όπου n είναι το πλήθος των παικτών στο δένδρο παικτών και για κάθε i, $1 \le i \le n$, m_i είναι το πλήθος των κόμβων του δένδρου εργασιών του i-οστού παίκτη στο δένδρο παικτών (σύμφωνα με την ενδοδιατεταγμένη διάσχιση), και για κάθε $j \in \{1,...,m_i\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της j-οστής εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, στο δένδρο εργασιών του i-οστού παίκτη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).

F

Γεγονός που υποδηλώνει τον τερματισμό της παρτίδας. Κατά το γεγονός αυτό θα πρέπει να πραγματοποιούνται οι ακόλουθες ενέργειες. Αρχικά, θα ελέγχετε το δένδρο παικτών και αν οι εξωγήινοι είναι περισσότεροι από τους ανθρώπους τότε η παρτίδα τερματίζει υπέρ των εξωγήινων. Εάν δεν υπάρχει κανένας εξωγήινος ή η ουρά προτεραιότητας των ολοκληρωμένων εργασιών περιέχει τόσες εργασίες όσες και ο συνολικός αριθμός εργασιών που έχουν ανατεθεί στους παίκτες, τότε η παρτίδα τερματίζει υπέρ των ανθρώπων. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, στην περίπτωση που νικήσουν οι εξωγήινοι, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

F
Aliens win.
DONE

Στην περίπτωση που νικήσουν οι άνθρωποι, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

F
Crewmates win.
DONE

X

Γεγονός τύπου Print Players που υποδηλώνει την εκτύπωση όλων των παικτών. Κατά το γεγονός αυτό τυπώνονται όλοι οι παίκτες με τα αναγνωριστικά τους και με την μεταβλητή που καθορίζει αν είναι εξωγήινοι ή όχι. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

X

Players = <pid$_1$:is_alien$_1$><pid$_2$:is_alien$_2$> … <pid$_n$:is_alien$_n$>

DONE

όπου n είναι ο αριθμός των κόμβων στο δέδνρο παικτών και για κάθε i ϵ{1,...,n}, <pid$_i$:is_alien$_i$> είναι το αναγνωριστικό του παίκτη και το αν είναι εξωγήινος αντίστοιχα, που αντιστοιχεί στον i-οστό κόμβο του δένδρου σύμφωνα με την ενδοδιατεταγμένη διάσχιση.

Y

Γεγονός τύπου Print All Tasks που υποδηλώνει την εκτύπωση όλων των εργασιών από το πίνακα κατακερματισμού γενικών εργασιών. Κατά το γεγονός αυτό τυπώνονται όλες οι εργασίες με τα αναγνωριστικά τους και με τον βαθμό δυσκολίας τους. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

Y

Index 0: <pid$_{0,1}$, difficulty$_{0,1}$>, <pid$_{0,2}$, difficulty$_{0,2}$>, ..., <aid$_{0,N0}$, difficulty$_{0,N0}$>

Index 1: <pid$_{1,1}$, difficulty$_{1,1}$ >, <aid$_{1,2}$, difficulty$_{1,2}$>, ..., <aid$_{1,N1}$, difficulty$_{0,N1}$>

...

Index n: <pid$_{n,1}$, difficulty$_{n,1}$>, <aid$_{n,2}$, difficulty$_{n,2}$>, ..., <aid$_{n,Nn}$, difficulty$_{n,Nn}$>

DONE

όπου n είναι το πλήθος των θέσεων του πίνακα κατακερματισμού γενικών εργασιών και για κάθε j, 0 ≤ j ≤ n-1, N j είναι το πλήθος των κόμβων στην αλυσίδα της θέσης j του πίνακα κατακερματισμού, όπως και για κάθε i, 1 ≤ i ≤ N j , <tid j,i > είναι το αναγνωριστικό της εργασίας που αντιστοιχεί στον i-οστό κόμβο της αλυσίδας που βρίσκεται στην θέση j του πίνακα κατακερματισμού.

Z

Γεγονός τύπου Print Completed Tasks που υποδηλώνει την εκτύπωση όλων των εργασιών που υπάρχουν στην ουρά προτεραιότητας μέγστων των ολοκληρωμένων εργασιών. Κατά το γεγονός αυτό τυπώνονται όλες οι εργασίες με τα αναγνωριστικά τους και το βαθμό δυσκολίας τους που υπάρχουν στην ουρά προτεραιότητας. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

Z

Completed Tasks = <tid$_1$,difficulty$_1$> <tid$_2$,difficulty$_2$> … <tid$_n$,difficulty$_n$>

DONE

όπου n είναι ο αριθμός των στοιχείων στην ουρά προτεραιότητας ολοκληρωμένων εργασιών και για κάθε i ϵ{1,...,n}, <tid$_i$> είναι το αναγνωριστικό της εργασίας που αντιστοιχεί στον i-οστό στοιχείο του πίνακα.

U

Γεγονός τύπου Print Tasks που υποδηλώνει την εκτύπωση όλων των παικτών και τα δένδρα εργασιών τους. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

U

$Player_1$ = <$tid_{1,1}$,$difficulty_{1,1}$>,<$tid_{1,2}$,$difficulty_{1,2}$> … <$tid_{1,mn}$,$difficulty_{1,mn}$>

$Player_2$ = <$tid_{2,1}$,$difficulty_{2,1}$>,<$tid_{2,2}$,$difficulty_{2,2}$> … <$tid_{2,mn}$,$difficulty_{2,mn}$>

…

$Player_n$ = <$tid_{n,1}$,$difficulty_{n,1}$>,<$tid_{n,2}$,$difficulty_{n,2}$>… <$tid_{mn,mn}$,$difficulty_{mn,mn}$>

DONE

όπου n είναι το πλήθος των παικτών στο δένδρο παικτών και για κάθε i, $1 \leq i \leq n$, m_i είναι το πλήθος των κόμβων του δένδρου εργασιών του i-οστού παίκτη στο δένδρο παικτών (σύμφωνα με την ενδοδιατεταγμένη διάσχιση), και για κάθε $j \in \{1,...,m_i\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της j-οστής εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, στο δένδρο εργασιών του i-οστού παίκτη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση).

Μικρό Bonus [5%]

Απελευθερώστε τη μνήμη που έχετε δεσμεύσει (στοιχεία όλων των δομών που έχετε δημιουργήσει) πριν τον τερματισμό του προγράμματός σας.

Δομές Δεδομένων

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

```
struct Player {
    int pid;
    int is_alien;
    int evidence;
    struct Player *parrent;
    struct Player *lc;
    struct Player *rc;
    struct Tasks *tasks;
};

struct Task {
    int tid;
    int difficulty;
    int lcnt;
    struct Task *lc;
    struct Tasks *rc;
};

struct HT_Task {
    int tid;
    int difficulty;
    struct Task *next;
};

int primes_g[650];
unsigned int max_tasks_g;
unsigned int max_tid_g;

struct General_Tasks_HT {
    int count;
    struct HT_Task **tasks;
}
```

```
struct General_Tasks_HT general_tasks_ht;

struct Completed_Tasks_PQ {
    int size;
    struct HT_Task tasks[];
};

struct Completed_Tasks_PQ completed_tasks_pq;
```

Βαθμολόγηση

P	8
T	10
D	10
I	10
E	10
W	5
S	15
V	7
G	15
F	2
X	2
Y	2
Z	2
U	2