HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο - Ακαδημαϊκό Έτος 2020-2021
Διδάσκουσα: Παναγιώτα Φατούρου
Προγραμματιστική Εργασία - 1ο Μέρος

Ημερομηνία Παράδοσης: Δευτέρα, 16 Νοεμβρίου 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.

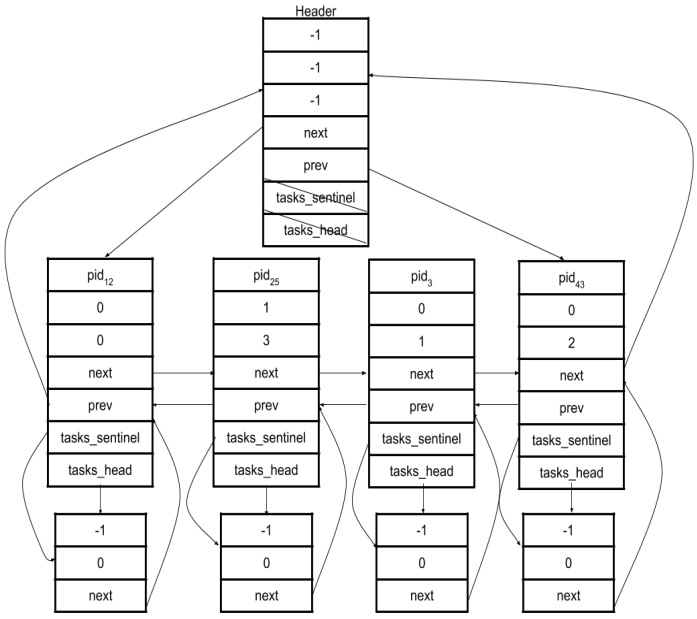
Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Κάθε παίκτης (player), εξωγήινος (impostor) ή άνθρωπος (crewmate), περιγράφεται από ένα μοναδικό αναγνωριστικό. Πληροφορίες για τους παίκτες αποθηκεύονται σε μία **διπλά συνδεδεμένη κυκλική λίστα με κόμβο φρουρό (Σχήμα 1), που λέγεται λίστα παικτών.** Κάθε στοιχείο της λίστας 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 στη λίστα, θεωρούμε ότι οι άλλοι παίκτες δεν έχουν πρόσβαση στο πεδίο is_alien κάθε παίκτη και άρα δεν μπορούν να ανακαλύψουν από εκεί, αν ο παίκτης είναι εξωγήινος ή άνθρωπος.

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

- **pid**: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον παίκτη.
- **is_alien**: Flag (τύπου int) που χαρακτηρίζει εάν ο παίκτης είναι εξωγήινος (impostor) ή άνθρωπος (crewmate).
- **evidence**: Ακέραιος που περιγράφει το βαθμό κατά τον οποίο ένας παίκτης θεωρείται ύποπτος (τύπου int).
- **prev**: Δείκτης (τύπου *struct Player*) που δείχνει στον προηγούμενο κόμβο της λίστας παικτών.
- **next**: Δείκτης (τύπου *struct Player*) που δείχνει στον επόμενο κόμβο της λίστας παικτών.
- **tasks_head**: Δείκτης (τύπου *struct Task*) που δείχνει στο πρώτο στοιχείο μιας λίστας με κόμβο φρουρό (που ονομάζεται λίστα εργασιών του παίκτη και περιγράφεται στη συνέχεια), η οποία περιέχει πληροφορίες για τις εργασίες (tasks) που θα ανατεθούν στον παίκτη.
- **tasks_sentinel**: Δείκτης (τύπου *struct Task*) που δείχνει στον κόμβο φρουρό της λίστας εργασιών του παίκτη.



Σχήμα 1. Η λίστα παικτών που είναι διπλά συνδεδεμένη, κυκλική, με κόμβο φρουρό και μη ταξινομημένη. Το δεύτερο πεδίο δείχνει αν ένας παίκτης είναι εξωγήινος ή όχι και το τρίτο πεδίο φανερώνει κατά πόσο ύποπτος είναι ένας παίκτης. Για λόγους απλοποίησης του σχήματος, θεωρούμε ότι το πεδίο tasks έχει την τιμή NULL σε όλα τα

στοιχεία της παραπάνω λίστας.

Ο κάθε παίκτης έχει διάφορες εργασίες (tasks) να ολοκληρώσει. Έτσι, για κάθε παίκτη υπάρχει μια **απλά συνδεδεμένη ταξινομημένη λίστα με κόμβο φρουρό**, **που λέγεται λίστα εργασιών του παίκτη.** Για κάθε εργασία του παίκτη υπάρχει ένα μοναδικό αναγνωριστικό (tid) και ένας βαθμός δυσκολίας (difficulty). Η λίστα εργασιών κάθε παίκτη είναι ταξινομημένη ως προς το βαθμό δυσκολίας της κάθε εργασίας που περιέχει.

Όλες οι εργασίες που πρέπει να πραγματοποιηθούν για να νικήσουν οι άνθρωποι, εισάγονται αρχικά σε μία **απλά συνδεδεμένη λίστα ,** η οποία είναι επίσης ταξινομημένη βάσει του βαθμού δυσκολίας. Η λίστα αυτή ονομάζεται **γενική λίστα εργασιών** (**Σχήμα 2**). Αφού όλες οι εργασίες εισαχθούν στη γενική λίστα εργασιών, θα μοιραστούν τελικά στις λίστες εργασιών των παικτών (**Σχήμα 3**).

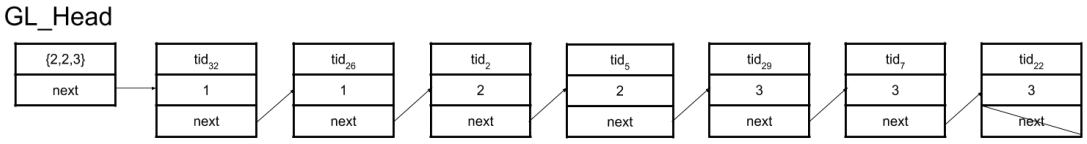
Κάθε κόμβος της γενικής λίστας εργασιών αλλά και της λίστας εργασιών ενός συγκεκριμένου παίκτη, αποτελεί μία εγγραφή τύπου struct Task με τα ακόλουθα στοιχεία:

- **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 Task*) που δείχνει στον επόμενο κόμβο.

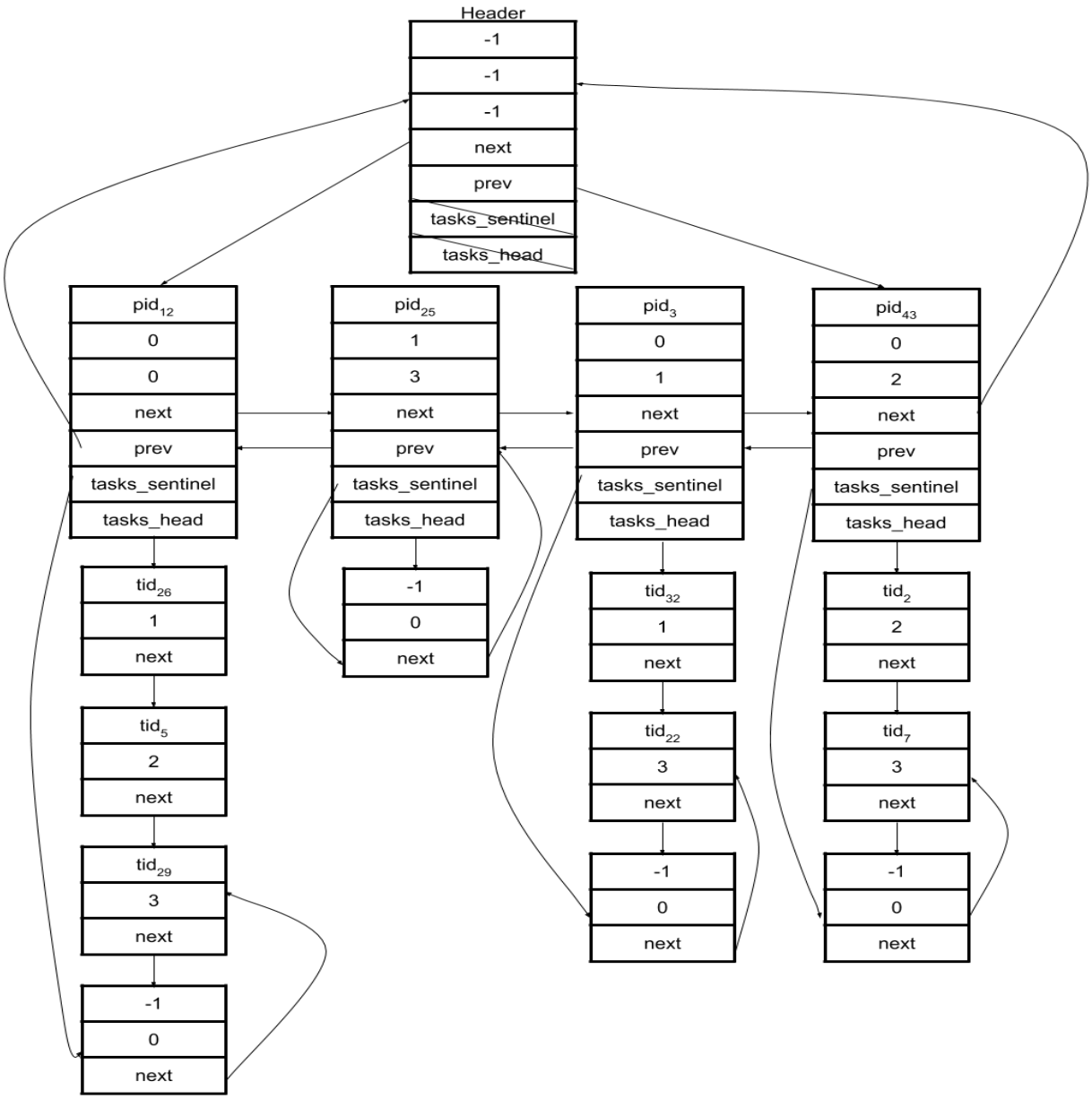
Υπάρχει ένα struct, τύπου General_List που περιέχει τα εξής πεδία:

- **GL**: Δείκτης σε struct τύπου Task που δείχνει στο πρώτο στοιχείο της γενικής λίστας εργασιών.
- **tasks_count[3]**: Πίνακας τριών ακεραίων. Το count[i] μετράει πόσες εργασίες βαθμού (i+1) υπάρχουν στη γενική λίστα εργασιών.

Υπάρχει και μια καθολική μεταβλητή Total_Tasks που αποθηκεύει το συνολικό πλήθος των εργασιών που αποθηκεύονται στη λίστα εργασιών.



Σχήμα 2. Η γενική λίστα εργασιών. Πρόκειται για μια απλά συνδεδεμένη λίστα που είναι ταξινομημένη ως προς τη δυσκολία της κάθε εργασίας.

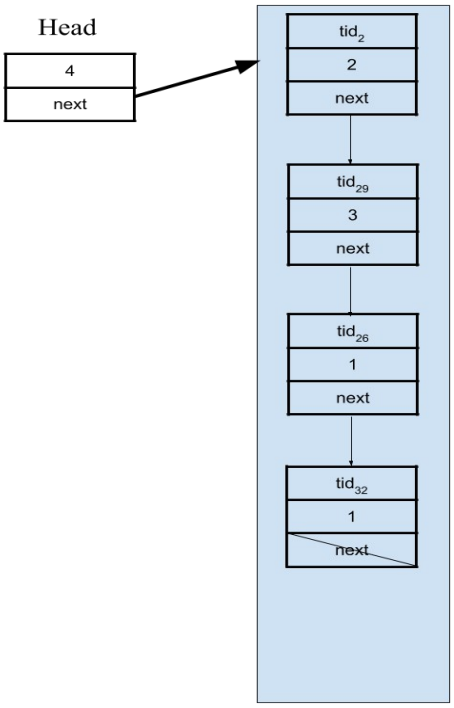


Σχήμα 3. Η διπλά συνδεδεμένη κυκλική λίστα με κόμβο φρουρό παικτών, η οποία είναι μη-ταξινομημένη, μετά την ανάθεση εργασιών στους παίκτες. Παρατηρήστε ότι η λίστα των εργασιών κάθε παίκτη είναι ταξινομημένη όπως και η γενική λίστα εργασιών με βάση τη δυσκολία της κάθε δουλειάς.

Αφότου έχουν καταχωρηθεί οι δουλειές στους ανθρώπους (μέλη του πληρώματος – creawmates), τότε εκείνοι αρχίζουν να τις υλοποιούν έτσι ώστε να καταφέρουν να νικήσουν την παρτίδα. Κάθε φορά που κάποιος άνθρωπος ολοκληρώσει μία εργασία τότε αυτή αφαιρείται από τη λίστα των εργασιών του και εισάγεται σε μία **στοίβα ολοκλήρωσης εργασιών** (**Σχήμα 4**). Κάθε κόμβος της στοίβας αποτελεί μία

εγγραφή τύπου struct Task (δηλαδή έχει ίδιου τύπου στοιχεία με τη γενική λίστα εργασιών και τις λίστες εργασιών των παικτών). Υπάρχει ένα struct, τύπου Head_Completed_Task_Stack που περιέχει τα εξής πεδία:

- **Head**: Δείκτης σε struct τύπου Task που δείχνει στο κορυφαίο στοιχείο της στοίβας.
- **count**: Ακέραιος που αποθηκεύει το πλήθος των στοιχείων της στοίβας.



Σχήμα 4. Η στοίβα με τις ολοκληρωμένες εργασίες. Αν γεμίσει 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 στη λίστα 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). Η λίστα εργασιών του παίκτη αρχικοποιείται να περιέχει μόνο τον

κόμβο φρουρό της. Θεωρήστε ότι το <pid> είναι μοναδικό (δηλαδή όλα τα <pid> που σχετίζονται με γεγονότα τύπου P στο test_file είναι διαφορετικά). Υλοποιήστε την εισαγωγή με τον πιο αποτελεσματικό τρόπο. 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 στη λίστα παικτών και για κάθε i ϵ{1,...,n}, <pid_i> είναι το αναγνωριστικό του παίκτη που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

T <tid> <difficulty>

Γεγονός τύπου Insert Task που υποδηλώνει τη δημιουργία μίας εργασίας με αναγνωριστικό <tid> και βαθμό δυσκολίας <difficulty>. Κατά το γεγονός αυτό, ένα νέο στοιχείο, που θα αντιστοιχεί σ' αυτήν την εργασία, θα εισάγεται στη γενική λίστα εργασιών. Το πεδίο <pid> του νέου στοιχείου είναι αρχικά απροσδιόριστο (θα προσδιοριστεί μέσω άλλου γεγονότος αργότερα). Επίσης, τα κατάλληλα πεδία του struct Genenal_List θα πρέπει να ενημερώνονται με το σωστό τρόπο. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

T <tid><difficulty>

Tasks = <tid_1,$difficulty_1$><tid_2,$difficulty_2$> … <tid_n,$difficulty_n$>

DONE

όπου n είναι ο αριθμός των κόμβων στη γενική λίστα εργασιών και για κάθε i ϵ{1,...,n}, <tid_i> είναι το αναγνωριστικό της εργασίας που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

D

Γεγονός τύπου Distribute Tasks που υποδηλώνει την ανάθεση εργασιών στους παίκτες. Κατά το γεγονός αυτό, θα πρέπει να μοιράζονται οι εργασίες που βρίσκονται στη γενική λίστα εργασιών στους παίκτες ακολουθώντας έναν αλγόριθμο εκ περιτροπής (round robin). Συγκεκριμένα, η διεργασία i στη λίστα εργασιών θα πρέπει να ανατίθεται στον παίκτη (i % n) όπου n είναι το πλήθος των στοιχείων στη λίστα παικτών. Το γεγονός αυτό θα πρέπει να έχει χρονική πολυπλοκότητα O(m), όπου m είναι το πλήθος των στοιχείων στη γενική λίστα εργασιών. Για να πετύχετε αυτή τη χρονική πολυπλοκότητα, χρησιμοποιήστε το next δείκτη του κόμβου φρουρού για να θυμάστε το τελευταίο στοιχείο στη λίστα εργασιών του εκάστοτε παίκτη. 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 για κάθε i, 1≤i≤ n, m_i είναι το πλήθος των κόμβων της λίστας των εργασιών του i-οστού παίκτη, και για κάθε j∈{1,...,m_i}, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της κάθε εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των εργασιών του i-οστού παίκτη.

I <pid> <difficulty>

Γεγονός τύπου Implement Task, που υποδηλώνει ότι ο παίκτης με το αναγνωριστικό <pid>, υλοποιεί μία εργασία με βαθμό δυσκολίας <difficulty>. Αν δεν υπάρχει κάποια εργασία με αυτό το βαθμό δυσκολίας, τότε ο παίκτης υλοποιεί μια εργασία από εκείνες που έχουν τη μεγαλύτερη δυσκολία στη λίστα εργασιών του. Κατά το γεγονός αυτό θα πρέπει να πραγματοποιούνται οι ακόλουθες ενέργειες. Αρχικά, θα αναζητάτε τον συγκεκριμένο παίκτη στην λίστα παικτών. Εφόσον έχετε βρει τον συγκεκριμένο παίκτη, θα αναζητάτε μια εργασία w βαθμού δυσκολίας <difficulty> στη λίστα εργασιών του παίκτη. Στη συνέχεια, θα διαγράφετε την w από τη λίστα εργασιών του παίκτη <pid> (έτσι ώστε η λίστα να παραμείνει ταξινομημένη και μετά το πέρας του γεγονότος) και θα εισάγεται έναν κόμβο για την w στη στοίβα με τις υλοποιημένες εργασίες (ενημερώνοντας σωστά και το πεδίο count του struct Stack). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

I<pid><difficulty>

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

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

…

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

DONE

όπου για κάθε i, 1≤i≤ n, mi είναι το πλήθος των κόμβων της λίστας των εργασιών του i-οστού παίκτη, και για κάθε j∈{1,...,mi}, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της κάθε εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των εργασιών του i-οστού παίκτη.

E <pid>

Γεγονός τύπου Eject Player που υποδηλώνει το άγγιγμα του παίκτη με αναγνωριστικό <pid> 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> αφαιρείται από τη λίστα των παικτών και όλες οι εργασίες του μεταφέρονται στον παίκτη ο οποίος έχει τις λιγότερες εργασίες εκείνη την στιγμή. Ο παίκτης <pid_min> με τις λιγότερες εργασίες θα βρίσκεται ως εξής: Θα πρέπει να διασχίσετε τη λίστα των παικτών και για κάθε παίκτη να μετράτε πόσα στοιχεία έχει η λίστα εργασιών του, κρατώντας έναν βοηθητικό δείκτη στην αρχή εκείνης της λίστας εργασιών που έχει βρεθεί να έχει τα λιγότερα στοιχεία την εκάστοτε χρονική στιγμή. Η εύρεση του <pid_min> πρέπει επομένως να πραγματοποιείται με μια διάσχιση της λίστας παικτών (και των λιστών εργασιών του κάθε παίκτη). Η

λίστα εργασιών του παίκτη <pid_min> πρέπει να παραμείνει ταξινομημένη μετά την συγχώνευσή της με τη λίστα εργασιών του παίκτη <pid>. Αφότου βρεθεί ο <pid_min>, η συγχώνευση των δύο λιστών θα πρέπει να εκτελείται σε χρόνο $O(n_1+n_2)$, όπου n1 είναι το πλήθος στοιχείων στη λίστα εργασιών του παίκτη <pid1> και n2 είναι το πλήθος στοιχείων στη λίστα εργασιών του παίκτη <pid_min>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

E <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 για κάθε i, 1≤i≤ n, mi είναι το πλήθος των κόμβων της λίστας των εργασιών του i-οστού παίκτη, και για κάθε j∈{1,...,mi}, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της κάθε εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των εργασιών του i-οστού παίκτη.

W <pid> <pid_a> <number_witnesses>

Γεγονός τύπου Witness Ejection που υποδηλώνει το άγγιγμα του παίκτη με αναγνωριστικό <pid> από τον εξωγήινο με αναγνωριστικό <pid_a> και την αφαίρεσή του παίκτη <pid> από το διαστημόπλοιο. Ωστόσο, στο γεγονός αυτό, την αφαίρεση του παίκτη <pid> την βλέπουν <number_witnesses> άνθρωποι (crewmates). Όπως και στο γεγονός τύπου E, κατά το γεγονός αυτό, ο παίκτης με αναγνωριστικό <pid> αφαιρείται από τη λίστα των παικτών και όλες οι εργασίες του μεταφέρονται στον παίκτη ο οποίος έχει τις λιγότερες εργασίες εκείνη την στιγμή. Ωστόσο, θα πρέπει να γίνει επιπρόσθετα το εξής: να διατρέξετε την λίστα των παικτών και να βρείτε τον εξωγήινο με αναγνωριστικό <pid_a> και να ενημερώσετε το πεδίο evidence για αυτόν προσθέτοντας τον αριθμό <number_witnesses>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

W <pid> <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 για κάθε i, 1 ≤ i ≤ n, mi είναι το πλήθος των κόμβων της λίστας των εργασιών του i-οστού παίκτη και $evidence_i$ είναι , και για κάθε j∈{1,...,mi}, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της κάθε εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των εργασιών του i-οστού παίκτη.

S <number_of_tasks><pid>

Γεγονός τύπου Sabbotage, το οποίο υποδηλώνει ότι ένας εξωγήινος αφαιρεί <number_of_tasks> εργασίες από τη στοίβα ολοκλήρωσης εργασιών και τις ξαναμοιράζει στους παίκτες. Κατά το γεγονός αυτό, πραγματοποιούνται <number_of_tasks> pop από την στοίβα ολοκλήρωσης εργασιών (ενημερώνοντας κατάλληλα το πεδίο count του struct Stack). Κάθε μια εργασία που αφαιρείται από τη στοίβα, ανατίθεται σε κάποιον παίκτη ακολουθώντας τον εξής αλγόριθμο: Αρχικά, αναζητούμε τον παίκτη με αναγνωριστικό <pid> στη λίστα παικτών. Η πρώτη εργασία ανατίθεται στον παίκτη p που βρίσκεται ⌊number_of_tasks/2⌋ θέσεις πιο αριστερά από τον παίκτη με αναγνωριστικό <pid>. Αν ο παίκτης p είναι εξωγήινος τότε, η εργασία ανατίθεται στον πρώτο κόμβο στα δεξιά του p που δεν είναι εξωγήινος. Κάθε επόμενη εργασία ανατίθεται στους επόμενους κόμβους του p (ακολουθώντας δείκτες next) που δεν είναι εξωγήινοι. Με τον τρόπο αυτό, οι ⌊number_of_tasks/2⌋ προηγούμενοι κόμβοι του p και τουλάχιστον οι ⌊number_of_tasks/2⌋ επόμενοι κόμβοι του p θα εξεταστούν για να τους ανατεθεί από μια εργασία στον καθένα. 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 για κάθε i, $1 \le i \le n$, mi είναι το πλήθος των κόμβων της λίστας των εργασιών του i-οστού παίκτη, και για κάθε $j \in \{1,...,mi\}$, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της κάθε εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των εργασιών του i-οστού παίκτη.

V <pid> <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> 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 στη λίστα παικτών αυξάνει κατά <vote_evidence>. Στη συνέχεια, πρέπει να διατρέξετε την λίστα των παικτών, να βρείτε τον παίκτη με το μεγαλύτερο evidence και να τον αφαιρέσετε. Η λίστα εργασιών του παίκτη που αποβάλλεται από το διαστημόπλοιο θα ανατεθεί στον παίκτη με τις λιγότερες εργασίες (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> <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 για κάθε i, $1 \leq i \leq n$, mi είναι το πλήθος των κόμβων της λίστας των εργασιών του i-οστού παίκτη και $evidence_i$ είναι , και για κάθε j∈{1,...,mi}, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της κάθε εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των εργασιών του i-οστού παίκτη.

G

Γεγονός τύπου Give Away Work, το οποίο υποδηλώνει τη μεταφορά εργασιών από έναν από τους παίκτες που έχουν τις περισσότερες εργασίες σε έναν από τους παίκτες που έχουν τις λιγότερες εργασίες (ή σ' έναν από αυτούς που έχει υλοποιήσει όλες τις εργασίες του και δεν είναι εξωγήινος). Κατά το γεγονός αυτό θα διατρέξετε τη λίστα των παικτών, θα βρείτε τον παίκτη <pmin> με τις λιγότερες εργασίες στη λίστα εργασιών του και τον παίκτη <pmax> με τις περισσότερες εργασίες στη λίστα εργασιών του (ακολουθώντας έναν αλγόριθμο αντίστοιχο με εκείνον που συζητήθηκε στο γεγονός Ε). Εφόσον πραγματοποιηθεί αυτό, θα μεταφέρετε τις μισές εργασίες, εκείνες με τον μικρότερο βαθμό δυσκολίας, που ανήκουν στον παίκτη <pmax>, στον παίκτη <pmin>. Ο αλγόριθμός σας θα πρέπει να εκτελείται με τον πιο αποτελεσματικό τρόπο που μπορείτε να σκεφτείτε. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

G
$Player_1$ = <$tid_{1,1}$,$difficulty_{1,1}$>,<$tid_{1,2}$,$difficulty_{1,2}$> … <$tid_{1,mn}$,$difficulty_{1,mn}$>
$Player_2$ = <$tid_{2,1}$,$difficulty_{2,1}$>,<$tid_{2,2}$,$difficulty_{2,2}$> … <$tid_{2,mn}$,$difficulty_{2,mn}$>
…
$Player_n$ = <$tid_{n,1}$,$difficulty_{n,1}$>,<$tid_{n,2}$,$difficulty_{n,2}$>… <$tid_{mn,mn}$,$difficulty_{mn,mn}$>
DONE

όπου για κάθε i, 1≤i≤ n, mi είναι το πλήθος των κόμβων της λίστας των εργασιών του i-οστού παίκτη, και για κάθε j∈{1,...,mi}, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της κάθε εργασίας και ο βαθμος δυσκολίας της αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των εργασιών του i-οστού παίκτη.

F

Γεγονός που υποδηλώνει τον τερματισμό της παρτίδας. Κατά το γεγονός αυτό θα πρέπει να πραγματοποιούνται οι ακόλουθες ενέργειες. Αρχικά, θα ελέγχετε τη λίστα των παικτών κι αν οι εξωγήινοι είναι περισσότεροι από τους ανθρώπους τότε η παρτίδα τερματίζει υπέρ των εξωγήινων. Εάν δεν υπάρχει κανένας εξωγήινος ή η στοίβα των 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 στη λίστα παικτών και 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 της λίστας αυτής.

Y

Γεγονός τύπου Print Tasks που υποδηλώνει την εκτύπωση όλων των εργασιών από τη γενική λίστα 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

List_Tasks = <tid$_1$,difficulty$_1$><tid$_2$,difficulty$_2$> … <tid$_n$,difficulty$_n$>

DONE

όπου n είναι ο αριθμός των κόμβων στη γενική λίστα εργασιών και για κάθε i ϵ{1,...,n}, <tid$_i$> είναι το αναγνωριστικό της εργασίας που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

Z

Γεγονός τύπου Print Stack που υποδηλώνει την εκτύπωση όλων των εργασιών που υπάρχουν στη στοίβα ολοκλήρωσης εργασιών. Κατά το γεγονός αυτό τυπώνονται όλες οι εργασίες με τα αναγνωριστικά τους και το βαθμό δυσκολίας τους που υπάρχουν στη στοίβα. Μετά το πέρας της εκτέλεσης του γεγονότος αυτού, η στοίβα θα πρέπει να παραμένει αναλλοίωτη (με όσα στοιχεία περιείχε και πριν γίνει η εκτύπωση των στοιχείων της). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

Z

Stack_Tasks = <tid$_1$,difficulty$_1$> <tid$_2$,difficulty$_2$> … <tid$_n$,difficulty$_n$>

DONE

όπου n είναι ο αριθμός των κόμβων στη λίστα εργασιών και για κάθε i ϵ{1,...,n}, $<tid_i>$ είναι το αναγνωριστικό της εργασίας που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

U

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

U

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

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

…

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

DONE

όπου για κάθε i, 1≤i≤ n, mi είναι το πλήθος των κόμβων της λίστας των εργασιών του i-οστού παίκτη, και για κάθε jϵ{1,...,mi}, $tid_{i,j}$ και $difficulty_{i,j}$ είναι το αναγνωριστικό της κάθε εργασίας και ο βαθμός δυσκολίας της αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των εργασιών του i-οστού παίκτη.

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 Players{
  int pid;
  int is_alien;
  int evidence;
  struct Players* prev;
  struct Players* next;
  struct Tasks* tasks_head;
  struct Tasks* tasks_sentinel;
};

struct Tasks{
  int tid;
  int difficulty;
  struct Tasks* next;
};

struct Head_GL{
    int tasks_count[3];
    struct Tasks* head;
};

struct Head_Completed_Task_Stack{
  int count;
  struct Tasks *head;
};
```

Βαθμολόγηση

P	7

T	8
D	15
I	10
E	15
W	5
S	13
V	7
G	10
F	2
X	2
Y	2
Z	2
U	2