ΑΣΚΗΣΕΙΣ ΣΕΤ 3 - ΔΔΑ ΤΜ. ΠΛΗΡ. & ΤΗΛΕΠ.

ΤΕΛΙΚΗ ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΑΔΟΣΗΣ: 03/01/2020

Άσκηση 1. Γράψτε ένα πρόγραμμα που να δημιουργεί ένα απλό blockchain. Το blockchain είναι μια αλυσίδα από μπλοκς για τα οποία ισχύει ότι το hash του προηγούμενου μπλοκ καταγράφεται ως πληροφορία στο τρέχον μπλοκ. Υλοποιήστε το blockchain σύμφωνα με τις ακόλουθες οδηγίες:

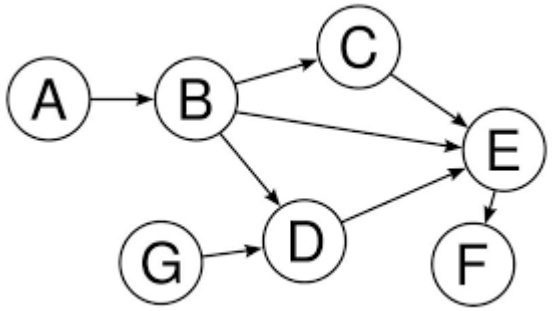
1. **Κάθε μπλοκ του blockchain να είναι ένα struct που να αποτελείται από τα εξής στοιχεία: id (τύπου size_t), timestamp (τύπου string), data (τύπου string), nonce (τύπου size_t) και previous_hash (τύπου size_t).**
2. **Να γράψετε συνάρτηση size_t hash_combined(block &a_block) που να επιστρέφει το hash ενός μπλοκ ως hash του λεκτικού που προκύπτει από τη συνένωση ως ένα λεκτικό των επιμέρους στοιχείων του μπλοκ. Για τον υπολογισμό του hash του λεκτικού να χρησιμοποιηθεί η std::hash.**
3. **Να γράψετε συνάρτηση void find_nonce(block &a_block, int difficulty) που να αλλάζει την τιμή του πεδίου nonce του a_block (ξεκινώντας από το 0 και δοκιμάζοντας διαδοχικά τιμές που αυξάνονται κατά 1) έτσι ώστε η hash τιμή του block να έχει τόσα συνεχόμενα μηδενικά στο τέλος όσα η τιμή της μεταβλητής difficulty.**
4. **Το αρχικό μπλοκ να έχει τα εξής στοιχεία: {0, <τρέχουσα ημερομηνία και ώρα>, "GENESIS BLOCK", <nonce>, 0} και να τοποθετείται σε μια std::list της STL. Η <τρέχουσα ημερομηνία και ώρα> να καταγράφεται ως YYYY-MM-DD HH:MM:SS. Το nonce να υπολογίζεται με difficulty=7.**
5. **Να συμπληρωθούν 7 επιπλέον μπλοκς έτσι ώστε το blockchain το οποίο θα έχει δημιουργηθεί με difficulty=7 να περιέχει πληροφορία αντίστοιχη με την ακόλουθη.**

id: 0 **ts: 2019-12-01 12:37:15** **data: GENESIS block** **nonce: 7705472** **p_hash: 0**	**id: 1** **ts: 2019-12-01 12:37:24** **data: Alice pays 10 euros to Bob** **nonce: 20662197** **p_hash: 7409222825570000000**	**id: 2** **ts: 2019-12-01 12:37:55** **data: Bob pays 5 euros to Carl** **nonce: 4180543** **p_hash: 14415237325170000000**	**id: 3** **ts: 2019-12-01 12:38:02** **data: Carl pays 10 euros to David** **nonce: 3124703** **p_hash: 9785420976540000000**
hash: 7409222825570000000	**hash: 14415237325170000000**	**hash: 9785420976540000000**	**hash:15500881473790000000**

id: 4 ts:2019-12-01 12:38:07 **data: David pays 2 euros to Alice** **nonce: 11311765** **p_hash: 15500881473790000000**	**id: 5** **ts: 2019-12-01 12:38:24** **data: Alice pays 2 euros to Bob** **nonce: 28602793** **p_hash: 17403203628000000000**	**id: 6** **ts: 2019-12-01 12:39:08** **data: Bob pays 5 euros to David** **nonce: 5567229** **p_hash: 847005160950000000**	**id: 7** **ts: 2019-12-01 12:39:17** **data: Carl pays 5 euros to Alice** **nonce: 6164283** **p_hash: 10509950750660000000**
hash:17403203628000000000	**hash:847005160950000000**	**hash:10509950750660000000**	**hash:11846123523500000000**

6. **Να γράψετε συνάρτηση bool check_valid_blockchain(list<block> &chain) που να επιστρέφει εάν το blockchain είναι έγκυρο ή όχι, εξετάζοντας την καταγεγραμμένη τιμή του previous_hash σε κάθε μπλοκ με την hash τιμή του προηγούμενου μπλοκ. Ελέγξτε την εγκυρότητα του blockchain.**
7. **Αλλάξτε το προτελευταίο block έτσι ώστε να περιέχει ως data το κείμενο «Bob pays 5000 euros to David» και ελέγξτε εκ νέου την εγκυρότητα του blockchain.**

Άσκηση 2. Υλοποιήστε τον αλγόριθμο του Kahn[1] για τοπολογική ταξινόμηση κατευθυνόμενων ακυκλικών γραφημάτων (Directed Acyclic Graphs). Θεωρείστε ότι τα γραφήματα καταγράφονται σε πίνακες γειτνίασης. Για παράδειγμα το γράφημα του σχήματος κάτω αριστερά καταγράφεται ως ο πίνακας κάτω δεξιά.



	A	B	C	D	E	F	G
A	0	1	0	0	0	0	0
B	0	0	1	1	1	0	0
C	0	0	0	0	1	0	0
D	0	0	0	0	1	0	0
E	0	0	0	0	0	1	0
F	0	0	0	0	0	0	0
G	0	0	0	1	0	0	0

Ειδικότερα, αναπτύξτε ένα πρόγραμμα που:

1. **Διαβάζει αρχείο στο οποίο είναι αποθηκευμένο ένα DAG.**
2. **Εφαρμόζει τον αλγόριθμο τοπολογικής ταξινόμησης του Kahn και καταγράφει τα αποτελέσματα σε αρχείο.**

Η εργασία θα πρέπει να παραδοθεί ως ένα zip αρχείο με όνομα e3_<arithmosmitroou>_<eponymo>_<onoma>.zip όπου στη θέση του <arithmosmitroou> θα αντικαταστήστε τον αριθμό μητρώου σας, στη θέση του <eponymo> το επώνυμό σας και στη θέση του <onoma> το όνομά σας, όλα με λατινικούς χαρακτήρες (π.χ. e3_12345_papadopoulos_ioannis.zip). Το zip αρχείο θα πρέπει να περιέχει:

- Πηγαίο κώδικα για την άσκηση 1 σε ένα αρχείο με όνομα ask1.cpp
- Την έξοδο που παράγει η άσκηση 1 σε ένα αρχείο με όνομα ask1.out
- Πηγαίο κώδικα για την άσκηση 2 σε ένα αρχείο με όνομα ask2.cpp
- Την έξοδο που παράγει η άσκηση 2 σε ένα αρχείο με όνομα ask2.out

[1] https://wikivisually.com/wiki/Topological_sorting