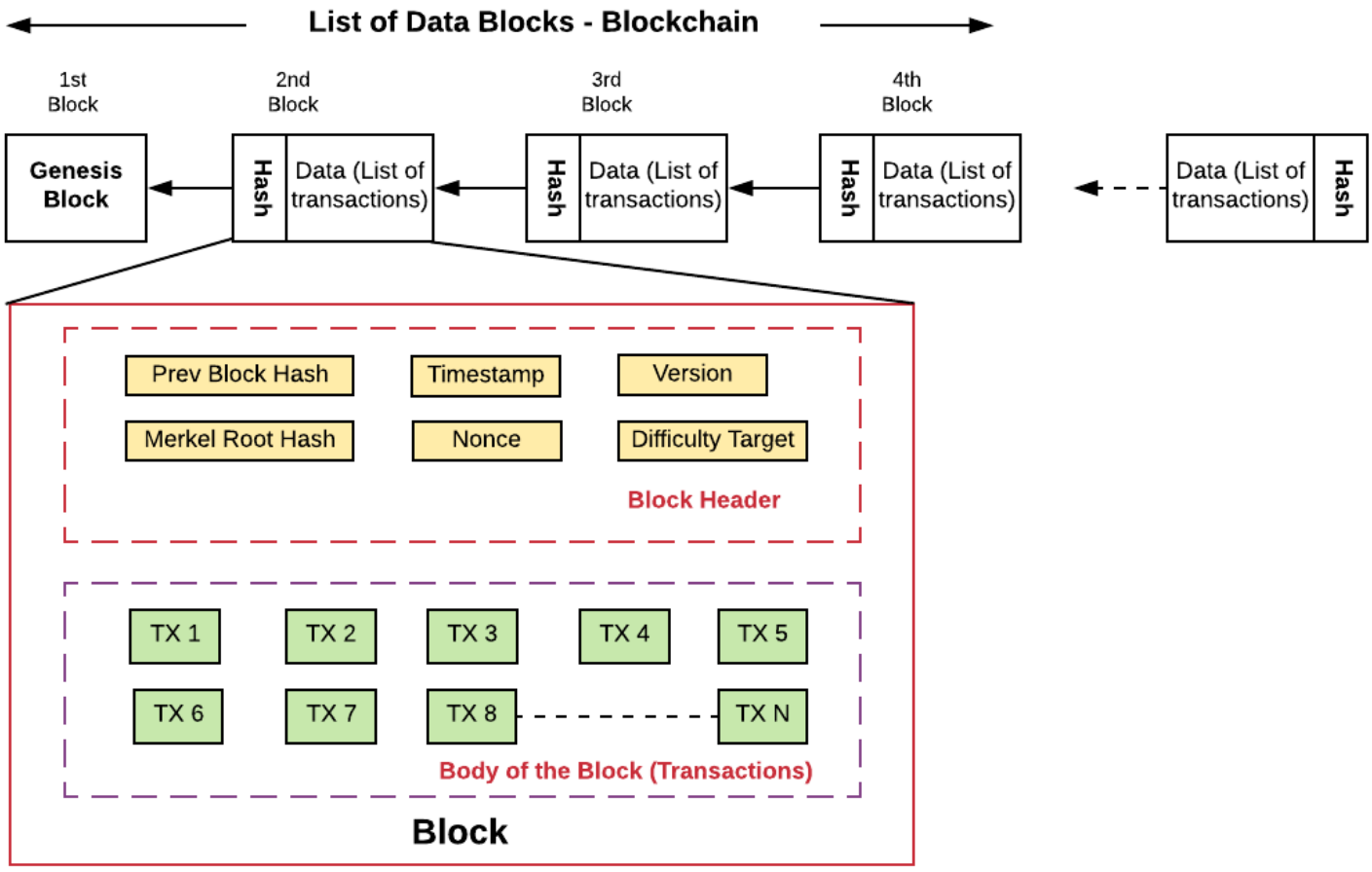
2-oji užduotis: Supaprastintos blokų grandinės (*blockchain*) kūrimas

Pagrindinis šios užduoties tikslas yra sukurti supaprastintą blokų grandinę (angl. *blockchain*), kurios galima duomenų sluoksnio struktūra yra pateikta žemiau:



Išskirtume kelis svarbius blokų grandinės aspektus:

- Blokų grandinė (*blockchain*) susidaro iš sąrašo vienas po kito einančių blokų, kurių kiekvienas susietas su prieš tai buvusio bloko hash'u (hash'uojama yra bloko antraštės (angl. *block header*) 6 pagrindiniai elementai).

- Kiekvieno iš *blokchain* grandinės bloko struktūra yra sudaryta iš dviejų esminių komponentų: **antraštės** (angl. *header*), kurią šiuo atveju sudaro:

    - Prieš tai buvusio bloko hash'as (`Prev Block Hash`)
    - Laiko žymė (`Timestamp`)
    - Blokams naudojamos duomenų struktūros versija (`Version`)
    - Visų bloko transakcijų binarinio Merkle medžio hash'as (`Merkel Root Hash`)
    - Atsitiktinio skaičiaus, kuris buvo panaudotas reikiamo sudėtingumo (nusakomo iš

eilės einančių nulių skaičiumi hash'o pradžioje) naujojo bloko hash'ui gauti (`Nonce`)
- Naujojo bloko hash'o radimo sudėtingumas (`Difficulty Target`)

ir **pagrindinės** (angl. *Body of the Block (Transactions)*) bloko duomenų dalies, kurią sudaro:

- į konkretų bloką įeinančios visos transakcijos.

Užduoties formuluotė

Sukurkite "centralizuotą" blokų grandinę (blockchain'ą) ir susimuliuokite blokų grandinės veikimą kuo natūralesnėmis sąlygomis. Norint tai pasiekti, preliminari veiksmų seka galėtų būti tokia (eilės tvarka gali kisti):

1. Sugeneruoti ~1000 tinklo vartotojų (aka *user*'ių), kurie turėtų bent tris atributus:
   - **vardą**,
   - **viešąjį *hash* raktą** (`public_key`)
   - tam tikros **valiutos atsitiktinį balansą** (pvz., nuo 100 iki 1000000 valiutos vienetų).
2. Sugeneruoti ~10000 naujų, į jokį bloką dar neįdėtų, transakcijų *pool*'ą, o transakcijų struktūra turėtų bent šiuos atributus:
   - **transakcijos ID** (kitų transakcijos laukų hash'as),
   - **siuntėjas** (jo viešasis raktas)
   - **gavėjas** (jo viešasis raktas)
   - **suma**

   Transakcijų struktūrą galite tobulinti savo nuožiūra pvz., vietoj sąskaitos modelio (angl. *account model*) galite adaptuoti UTXO modelį.
3. Iš transakcijų *pool*'o atsitiktinai pasirinkti 100-ą transakcijų, kurias bandysime įdėti į naują bloką. Tarsime, kad naujas blokas talpins apie 100 transakcijų. Reikiama bloko struktūra ir būtini atributai pateikti paveiksle aukščiau.
4. Realizuokite naujų blokų kasimo (angl. *mining*) *Proof-of-Work* (PoW) tipo procesą, kurio tikslas yra surasti naujam blokui hash'ą, tenkinantį `Difficulty Targer` reikalavimą, t.y., hash'o pradžioje esančių nulių tam tikrą skaičių. Nulių skaičius priklauso nuo Jūsų sukurtos hash funkcijos savybių ir efektyvumo. Paeksperimentuokite, kad tai neužtruktų per ilgai. Kaip matyti, bloko kasimui yra reikalingas transakcijų Merkle hash'as, kuris taip pat turi būti realizuotas (žr. detalizaciją versijų reikalavimuose).
5. Suradus reikiamų savybių naujo bloko hash'ą:

    - iš transakcijų pool'o ištrinkite į naują bloką priskirtas transakcijas;
    - "įvykdykite" transakcijas, t.y., atnaujinkite tinklo vartotojų balansus;
    - naują bloką pridėkite prie *blockchain* grandinės.
6. Kartoti **3-5** žingsnius tol, kol yra laisvų transakcijų. Galima būtų kartoti ir visus **1-5** žingsnius, tokiu būdu įtraukiant ir naujų vartotojų bei transakcijų kūrimą, o ciklą stabdyti naudojantis kitomis logiškomis sąlygomis.

Reikalavimai versijai (`v0.1`) (Terminas: 2022-10-27)

- Vadovaujantis užduoties formuluote, realizuokite supaprastintą "centralizuotą" blockchain'ą. Hash'avimo procesui naudokite Jūsų pirmąjam darbui sukurtą *hash* funkciją.
- Transakcijų ir naujų blokų kūrimo procesas turi būti matomas, t.y., output'inimas kūrimo metu, o ir turi būti sukurtos funkcijos, leidžiančios atspaudinti bet kurią transakciją ir bloką. Output'inimo detalumas/vizualumas turės tiesioginės įtakos balui, o kaip pvz.: https://www.blockchain.com/btc/block/1
- Šioje versijoje vietoj sudėtingesnės dvejetainio Merkle medžio hash'o realizacijos, galite paimti visų į naują bloką dedamų transakcijų hash'ą.
- Šiai versijai sukurkite releas'ą ir `README` faile detaliai aprašykite, kaip naudotis šia *blockchain*' versija. Jos atskirai atsiskaityti nereiks - reikės apginti galutinę versiją `v0.2`, tačiau bus žiūrima ar `v0.1` buvo realizuota laiku.
- **Kadangi blockchain'ams saugumas yra kertinis dalykas, todėl realizacijose turite tikslingai naudoti gerąsias OOP praktikas, kaip pvz., enkapsuliavimas, konstruktoriai, RAII idioma ir pan., t.y., kad nebūtų taip, kad privatus vartotojo raktas yra klasės `public` narys. Tą pristatymo metu turėsite akcentuoti.**

Reikalavimai versijai (`v0.2`) (Terminas: 2022-11-03)

- Tiems, kas `v0.1` nedarėte, *Merkle Root Hash* turi būti realizuotas pagal binarinį *Merkle Tree* veikimą.
- Antrame žingsnyje realizuokite transakcijų verifikavimą:
    - **Balanso tikrinimas**: Jeigu Jūsų sukurtoje transakcijoje vartotojas A siunčia tam tikrą pinigų sumą vartotojui B, tačiau jo balansas yra mažesnis negu siunčiama suma, tą transakciją reiktų ištrinti iš transakcijų pool'o/iš viso neįdėti į transakcijų pool'ą.
    - **Transakcijos hash'o tikrinimas**: Pool'o lygmenyje realizuokite validumo patikrinimą, t.y., ar tranksakcijos informacijos hash'as sutampa su transakcijos ID. Tokiu būdu įsitikinsite, kad transakcija nebuvo suklastota, kol ji "keliavo" iki transakcijų pool'o.
- Patobulinkite blokų kasimo (*mining*) procesą pagal tokią (ar panašią) logiką:
    - Pirmiausiai pagal aukščiau pateiktą aprašą 3-ame žingsnyje iš visų transakcijų *pool*'o,

sudarykite ne vieną, o penkis naujus potencialius blokus (kandidatus): 1A, 1B, 1C, 1D, 1E, sudarytus iš ~100 atsitiktinai pasirinktų tranksakcijų, kurios šiuose blokų kandidatuose gali persidengti.
    - Tuomet atsitiktinai pasirinkite vieną iš tų penkių blokų-kandidatų ir su juo tam tikrą fiksuotą laiką (pvz., iki 5 sek.) arba fiksuotą maksimalų hash'avimų bandymų skaičių (pvz., iki 100000) atlikite kasimo procesą kaip aprašyta užduoties 4-ame žingsnyje. Jeigu po to laiko, nebuvo šiam atsitiktinai pasirinktam blokui-kandidatui surastas reikiamų savybių hash'as, tenkinantis `Difficulty Targer` reikalavimą, tą procesą pakartokite su kitu atsitiktinai pasirinktu likusiu (vienu iš keturių, trijų ir t.t.) kandidatu.
    - Jeigu procesą pakartojus su visais penkiais blokais-kandidatais nebuvo surastas reikiamų savybių hash'as (naujas blokas nebuvo sukurtas), tuomet pailginkite kasimo procesą (pvz. iki 10 sek.) ar padidinkite maksimalų hash'ų bandymų skaičių (pvz. iki 200000) ir pakartokite aukščiau aprašytą procesą.

Tokiu būdu nuosekliame programos veikimo kontekste gana naiviai sumodeliuojame "decentralizuotą" naujų blokų kasimą.

Papildomos užduotys

1. Jei vietoje sąskaitos modelio (angl. *account model*) adaptuosite UTXO modelį, papildomai už tai galite uždirbti iki 0.5 balo.
2. Jei `v0.2` kasimo schemą realizuosite ne nuosekliai, o **lygiagrečiai**, už tai papildomai galite uždirbti iki 0.5 balo.