7. Ühetaoline tööaeg (`konstant`)

5 sek / 10 sek 100 punkti

Turvaliste süsteemide programmeerimisel on vaja olla ettevaatlik, et pahalased neisse sisse murda ei saaks. Üks oht on niinimetatud ajastusründed: tihtipeale kulutavad programmid erinevate sisenditega käivitades erineva hulga aega ning mõnel juhul on võimalik seda ajakulu ka üle interneti jälgida ja saada selle kaudu infot salajaste andmete kohta, mida programmid töötlevad. Näiteks proovivad informaatikavõistluste osalejad vahel niimoodi testide sisu ära arvata.

Viise, kuidas sellised ajastuse erinevused tekivad, on mitmesuguseid: kõige ilmsem on vast see, kui programm läbib eri sisenditega mõnda tsüklit erinev arv kordi. Olgu meil näiteks funktsioon, mis kontrollib, kas kaks sõne on võrdsed. Kui funktsioon kohe esimese erinevuse juures eitava vastuse tagastab, siis on piisavalt täpselt programmi tööaega mõõtes võimalik leida, mitmendal positsioonil sõnede esimene erinevus aset leidis.

Aga ka täpselt sama käskude jada täitvate programmide tööajad võivad erineda. Näiteks võib ühe moodsa AMD protsessori peal 64-bitiste arvude jagamine võtta 10–18 protsessori takti, sõltuvalt jagatavate väärtusest. Mõnedel väga vanadel protsessoritel võib ka korrutamise ja mõnikord isegi bitinihutamise jaoks kuluv aeg sõltuda argumentide väärtustest.

Veel üks viis, kuidas ajastuse erinevused tekivad, tuleneb protsessorite vahemälust. Vahemälus olevate andmete lugemine on märksa kiirem kui oleks samade andmete lugemine põhimälust. Paari lisateguri tõttu tähendab see tegelikult, et igasugune mälust andmete lugemine võib ründajale paljastada aadressi, millelt andmeid loeti. Seega ei tohi turvalises koodis näiteks massiivi „salajase“ väärtusega indekseerida.

Sinu ülesanne on lahendada mõned klassikalised informaatikaülesanded konstantajas, ehk nii, et paljastad ajastuse kaudu sisendi kohta võimalikult vähe informatsiooni. Täpsemalt ei tohi ajastusse lekkida midagi ülesande sisendi *väärtuste* kohta: sisendi *suuruse* lekkimine on praktiliselt vältimatu. Seega peab programmi tööaeg kõigi ühesuuruste sisenditega alati täpselt sama olema. Selles ülesandes loeme, et

- kõik sisendi väärtused on sajalased;
- igasugune aritmeetiline tehe salajase väärtusega annab tulemuseks salajase väärtuse;
- salajasi väärtusi ei tohi kasutada `if`-, `while`-, või `for`-käskude tingimustes;
- salajasi väärtusi ei tohi kasutada jagamise ega jäägi leidmise tehetes;
- salajasi väärtusi ei tohi kasutada massiivide indekseerimiseks.

Selles ülesandes on viis alamülesannet, mille püstitused on toodud järgmistes jaotistes.

Selles ülesandes saab lahendusi esitada **ainult keeltes C++ ja Python**. Lahenduses tuleb iga alamülesanne realiseerida eraldi funktsioonis. Lahenduse ühe täitmise ajal võib hindamissüsteem ühte funktsiooni korduvalt välja kutsuda.

Salajased väärtused on realiseeritud klassina `secret_int`, mis muidu käitub nagu tavaline märgita täisarv, kuid millel keelatud operatsioonide sooritamine annab veateate. `secret_int` on 32-bitine, kõik aritmetilised tehted sooritatakse mooduli 2^{32} järgi (näiteks $\texttt{0xffffffff} + 1 = 0$). Võrdlustehted `secret_int`idega tagastavad samuti `secret_int`i väärtusega 0 või 1 vastavalt võrdluse tulemusele.

Testimiskeskkonnas on antud näitefailid, kus vajalikud funktsioonid on juba kirjeldatud ja vaja on need vaid realiseerida. Lisaks võib lahenduse faili kirjutada ka oma funktsioone. Oma lahenduse oma arvutis testimiseks on ka hindamisprogrammi näide, mille sisendi ja väljundi kirjeldus on toodud allpool. Serveris on kasutusel teine hindamisprogramm, mis kontrollib ka lahenduse tagastatud vastuse õigsust. Oma lahenduse oma arvutis kompileerimiseks ja testimiseks:

- C++ keeles võib lahenduse failinimi olla suvaline. Näiteks `konstant.cpp` korral on kompileerimiseks käsk `g++ -o grader sample_grader.cpp konstant.cpp` ja käivitamiseks käsk `./grader`. Lisaks peab fail `secret_int.h` kompileerimise ajal samas kaustas olema.
- Pythonis peab lahenduse failinimi olema `konstant.py`, lahenduse käivitamiseks on käsk `python3 sample_grader.py` ja lisaks peab fail `secret_int.py` samas kaustas olema.

Pange tähele, et C++ keeles ei tohi realiseerimata funktsioone ära kustutada, kuna muidu tekib kompileerimisel viga ja lahendus ei saa punkte ka lahendatud alamülesannete eest.

Kuigi žürii on üritanud testimiskeskkonnas salajaste väärtuste semantika võimalikult turvaliselt realiseerida, võib siiski juhtuda, et süsteemi piirangutest on võimalik mööda hiilida. Lahendused, kus on `secret_int` väärtusi mittesihtotstarbeliselt kasutatud, punkte ei saa.

Näidishindamisprogramm loeb sisendi esimeselt realt alamülesande numbri S ($0 \leqslant S \leqslant 4$) ja testide arvu T. Edasi loetakse iga testi sisend (mille vorming sõltub alamülesandest), kutsutakse välja lahenduse funktsioon ja väljastatakse funktsiooni tagastatud väärtus.

$S = 0$. Massiivi indekseerimine

Antud on massiiv A pikkusega N ja täisarv i ($0 \leqslant i < N$). Leida $A[i]$.

```
// C++
secret_int ith_element(std::vector<secret_int> A, secret_int i);
# Python
def ith_element(A: List[secret_int], i: secret_int) -> secret_int:
```

Näidishindamisprogramm loeb sisendi esimeselt realt kaks täisarvu N ja i ning teiselt realt N täisarvu, milleks on massiivi A elemendid. Näidishindamisprogramm trükib väljundisse ainsale reale funktsiooni tagastatud väärtuse.

Näide.

Sisend	Väljund
0 2 3 2 1 2 3	3
3 0 4294967295 1 2	4294967295

Selles näites on 2 testi. Esimeses testis on massiiviks $[1, 2, 3]$ ja küsitakse selle elementi indeksiga 2, milleks on 3. Teises testis küsitakse massiivi $[4294967295, 1, 2]$ esimest elementi, milleks on 4294967295.

$S = 1$. Massiivi sorteerimine

Antud on massiiv A pikkusega N. Ülesandeks on A sorteerida, s.t. järjestada ümber mittekahanevasse järjekorda.

```
// C++
void sort_arr(std::vector<secret_int>& A);
# Python
def sort_arr(A: List[secret_int]):
```

Siin ülesandes tuleb massiivi A muuta kohapeal, midagi tagastama ei pea.

Näidishindamisprogramm loeb sisendi esimeselt realt täisarvu N ja teiselt realt N täisarvu, milleks on massiivi A elemendid. Näidishindamisprogramm trükib väljundi ainsale reale terve massiivi pärast lahendusfunktsiooni välja kutsumist.

Näide.

Sisend	Väljund
1 1 8 3 5 6 5 6 2 5 8	2 3 5 5 5 6 6 8

Selles näites on vaid üks test. Väljundiks on antud massiiv sorteerituna.

$S = 2$. Graafi sidususkomponentide arvu leidmine

Antud on N-tipuline graaf. Graaf on esitatud naabrusmaatriksina A. See tähendab, et $A[i][j]$ on 1, kui graafi tippude i ja j vahel on serv, ja 0, kui ei ole. On teada, et $A[i][j] = A[j][i]$ ja $A[i][i] = 0$. Leida, mitu sidususkomponenti selles graafis on.

```
// C++
secret_int connected_components(std::vector<std::vector<secret_int>> A);
# Python
def connected_components(A: List[List[secret_int]]) -> secret_int:
```

Näidishindamisprogramm loeb sisendi esimeselt realt täisarvu N ja järgmiselt N realt igalt N täisarvu — maatriksi A elemendid — ning väljastab funktsiooni tagastatud väärtuse.

Näide.

Sisend	Väljund
2 1 6 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0	3

Siin on üks test. Graafis on 6 tippu ning 3 sidususkomponenti: $\{0\}$, $\{1, 2\}$ ja $\{3, 4, 5\}$.

$S = 3$. Suunatud graafis teekonna leidmine

Antud on N-tipuline kaalutud suunatud graaf (mille tipud on nummerdatud 0 kuni $N-1$), mille servi kirjeldab maatriks A. Kui $A[i][j]$ on 0, siis tipust i tippu j serva ei lähe; vastasel juhul on $A[i][j]$ serva kaal (kõik kaalud on positiivsed). On teada, et $A[i][j] < 2^{32}/N$ ja $A[i][i] = 0$. Lisaks on antud täisarvud B ja C, kus $0 \leqslant B < N$ ja $0 \leqslant C < N$. Leida lühim (minimaalse kaaluga) teekond tipust B tippu C. On teada, et leidub vähemalt üks teekond tipust B tippu C.

```
// C++
std::tuple<secret_int, secret_int, std::vector<secret_int>>
graph_path(std::vector<std::vector<secret_int>> A,
    secret_int B, secret_int C);
```

```
# Python
def graph_path(A: List[List[secret_int]],
        B: secret_int, C: secret_int,
 ) -> Tuple[secret_int, secret_int, List[secret_int]]:
```

Tagastada tuleb kolmik (K, L, M), kus K on teekonna kogukaal, L läbitud tippude arv ja M massiiv läbitud tippudest. Kuna teekonna pikkust ei tohi massiivi pikkusena lekitada, siis võib massiiv M olla pikem kui vaja; hindamisprogramm loeb sellest vaid esimesed L elementi.

Näidishindamisprogramm loeb sisendi esimeselt realt 3 täisarvu N, B ja C. Järgmiselt N realt loetakse igalt N täisarvu: maatriksi A elemendid. Väljundi esimesele reale trükitakse arv K, teisele L, ja kolmandale massiivi M esimesed L elementi.

Näide.

Sisend	Väljund
3 1 4 0 3 0 3 0 9 0 0 2 4 0 2 0 3 0 0 0 0	7 3 0 1 3

Lühim teekond tipust 0 tippu 3 on tipust 0 tippu 1 (kaal 3) ja tipust 1 tippu 3 (kaal 4), kokku kaaluga 7 ja pikkusega 3.

$S = 4$. Algarvulisuse kontrollimine

Antud on positiivne täisarv N. Leida, kas N on algarv. Tagastada 1, kui on, või 0, kui ei ole.

Selles alamülesandes ei tohi üheski testis `secret_int`idega teha rohkem kui 10^6 tehet.

```
// C++
secret_int is_prime(secret_int N);
# Python
def is_prime(N: secret_int) -> secret_int:
```

Näidishindamisprogramm loeb sisendi ainsalt realt arvu N ja trükib väljundisse funktsiooni tagastatud väärtuse.

Näide.

Sisend	Väljund
4 2 4 11	0 1

Esimeses testis 4 ei ole algarv, seega vastus on 0. Teises testis 11 on algarv, seega vastus on 1.

Programmeerimise näide

`secret_int` väärtuste kasutamise kohta võib abiks olla järgnev näide, mis realiseerib turvaliselt sissejuhatuses mainitud sõnede võrdlemise funktsiooni (siin küll arvumassiivide võrdlemine).

```
// C++:
secret_int compare(std::vector<secret_int> a, std::vector<secret_int> b) {
    if (a.size() != b.size()) {
        return 0;
    }
    secret_int result = 1;
    for (int i = 0; i < a.size(); i++) {
        result &= (a[i] == b[i]);
    }
    return result;
}
# Python:
def compare(a: List[secret_int], b: List[secret_int]) -> secret_int:
    if len(a) != len(b):
        return secret_int(0)
    result = secret_int(1)
    for x, y in zip(a, b):
        result &= (x == y)
    return result
```

Hindamine

Selles ülesandes on testid jagatud gruppidesse. Iga grupi eest saavad punkte ainult need lahendused, mis läbivad **kõik** sellesse gruppi kuuluvad testid. Gruppides kehtivad järgmised lisatingimused:

1. (0 punkti) Näited.
2. (8 punkti) $S = 0$ (massiivi indekseerimine), $T \leqslant 10$, $N \leqslant 100\,000$.
3. (9 punkti) $S = 1$ (massiivi sorteerimine), $T \leqslant 10$, $N \leqslant 2000$.
4. (8 punkti) $S = 1$ (massiivi sorteerimine), $T \leqslant 10$, $N \leqslant 30\,000$.
5. (20 punkti) $S = 2$ (sidususkomponentide loendamine), $T \leqslant 10$, $N \leqslant 150$.
6. (25 punkti) $S = 3$ (graafis tee leidmine), $T \leqslant 10$, $N \leqslant 100$.
7. (10 punkti) $S = 4$ (algarvulisuse kontrollimine), $T \leqslant 500$, $N < 10^6$.
8. (20 punkti) $S = 4$ (algarvulisuse kontrollimine), $T \leqslant 500$, $N < 2^{32}$.