

1. Gyakorlati rész

Ez a dokumentum az SWSV vizsga gyakorlati részéhez kapcsolódó mintafeladatokat tartalmaz. Ezek a feladatok nem feltétlenül fedik le az összes lehetséges feladattípust, ezért érdemes az előadások anyagát is megnézni.

1.1. Forráskód statikus analízise

1. Statikus analízis

Adott a következő forráskód részlet, ahol az `ioread32()` függvény egy 32 bites, előjeles egész számot olvas be.

```
int64 a, b
1: a = ioread32()
2: if (a < 0) {
3:     b = -a
4: } else {
5:     b = a
6: }
```

- (a) (2 pont) Írja le, hogy absztrakt interpretációt használva az egyes kódsorok esetén a programváltozók értékei hogyan alakulnak (X_1, X_2, \dots, X_6)! Ahol konkrét érték nem ismert, ott szimbolikusan számoljon (felhasználva a korábbi értékeket)!

- (b) (2 pont) A program belépési pontjától kezdve (X_1) intervallumok segítségével terjessze végig a programváltozók értékészletét! Mi az b változó lehetséges értékészlete a 6. sorban (X_6)?

1.2. Specifikáció alapú tesztelés

2. Kombinatorikus tesztelés

Készülő alkalmazásunk elérhető iOS, Android és Windows platformokra. Windows esetén PC-ken is, míg a másik két platformon csak okostelefonon és tableten támogatott. Jelenleg magyarra és angolra van lefordítva.

- (a) (1 pont) Hány különböző konfigurációban lehetne tesztelni az alkalmazást? _____
- (b) (3 pont) Megelégszünk csak a páronkénti lefedettséggel. Mutasson egy olyan konfigurációhalmazt, ami minimális és kielégíti a páronkénti lefedettség kritériumát!

#	Platform	Eszköz	Nyelv
---	----------	--------	-------

3. Döntési táblák

Biztosítási díjat számoló alkalmazást készítünk. A díj az autó teljesítményén alapszik (500 Ft / kW). Ha a tulajdonosnak van családja vagy közalkalmazott, akkor 5% kedvezmény adható (de legfeljebb az egyik fajta kedvezmény számít csak). További 5% kedvezmény adható, ha a tulajdonosnak nem volt a tavalyi évben balesete (ezt a kedvezményt az előző kedvezmény után kell érvényesíteni). Azonban az összes kedvezmény figyelembe vétele után is legalább 25 000 Ft a biztosítási díj.

- (a) (3 pont) Készítsen egy döntési táblát a fenti specifikáció teszteléséhez.

--

- (b) (1 pont) Milyen konkrét teszteseteket hajtana végre a döntési tábla alapján?

--

1.3. Struktúra alapú tesztelés

4. Struktúra alapú tesztelés

Adott a következő forráskód részlet.

```
int function13(int a, bool b){
    int c = 10;
    if (b) c = -c;

    for (int i = 0; i < a; i++){
        c++;
    }

    return c;
}
```



- (a) (2 pont) Rajzoljuk fel a függvény vezérlési folyam gráfját (CFG) a függvény kódja mellé!
 (b) (2 pont) Adjunk meg egy tesztesetet, mely 100%-os utasítás lefedettséget garantál!

5. Struktúra alapú tesztelés

Adott a következő függvény (bal), és egy tesztbemenet-halmaz, amivel teszteljünk (jobb).

```
void checkParameters(bool a, int b, int c, bool d){
    if (!a || (b > 0)){
        error();
        return;
    }

    if ( (c == 0) && d && (b < -100) ){
        warning();
        return;
    }

    info();
    return;
}
```

#	a	b	c	d
T1	false	0	0	true
T2	true	0	1	false

- (a) (2 pont) Hány százalékos döntés lefedettséget érnek el a tesztek? (A számítást is kérjük mellékelni.)

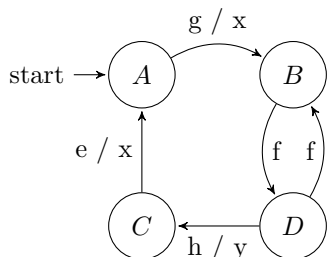
- (b) (2 pont) Hány százalékos feltétel lefedettséget érnek el a tesztek? (A számítást is kérjük mellékelni.)



1.4. Tesztgenerálás

6. Modellalapú tesztgenerálás

Adott a következő véges automata:



(a) (3 pont) Adjon egy olyan tesztkészletet, ami 100% átmenet lefedettséget ér el!

7. Szimbolikus végrehajtás

Adott a következő forráskód részlet.

```

public int Handler(State s, int v) {
    switch(s) {
        case Active:
            return 0;
        case Inactive:
            return -1;
        default:
            v = v + 1;
    }
    if(v > 10) throw new Exception();
    return v;
}
    
```

- (a) (3 pont) Rajzolja fel a Handler függvény szimbolikus végrehajtási fáját! Jelölje a kivételt előidéző lefutási útvonala(ka)t a többitől eltérően!
- (b) (1 pont) Milyen útvonal-feltétel (path condition) adódik így a kivételt előidéző útvonalon?

1.5. Regressziós tesztelés

8. Regresszióteszt-kiválasztás

Van egy webes áruház alkalmazásunk egy JavaScript felülettel, felhő-alapú készletnyilvántartás és termék ajánló szolgáltatásokkal és egy adatbázissal. Az egységtesztek a készletnyilvántartást (UT1) és az ajánló modult (U2) vizsgálják. Az integrációs tesztek a szolgáltatások API-jait hívják közvetlenül. Az első integrációs teszt (IT1) egy egyszerű vásárlást, a második (IT2) egy ajánlott termék vásárlását vizsgálja. A rendszerteszt (ST1) a webes felületen keresztül hívja meg a teljes rendszert.

A fejlesztőcsapat az ajánló modul algoritmusát módosította.

- (a) (1 pont) Adja meg a tesztek és a tesztelt komponensek közötti kapcsolatokat!

- (b) (2 pont) Oszályozza az egyes tesztek az adott módosítás viszonylatában!

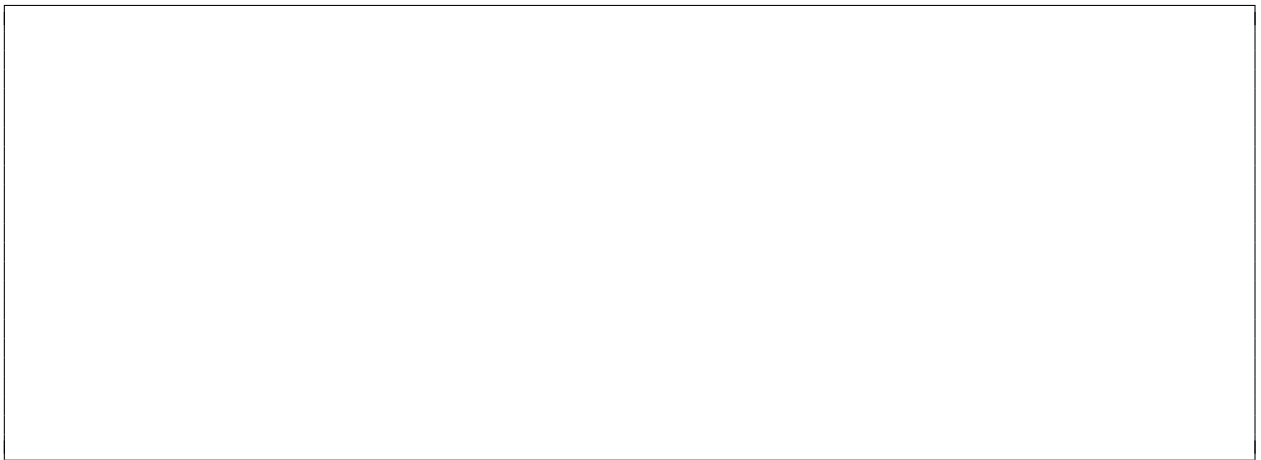
1.6. Szolgáltatásbiztonság analízise

9. Eseményfa

Egy automatikus vezérlésű vonaton két fedélzeti számítógép van. Amennyiben a fő számítógép meghibásodik, a tartalék számítógép veszi át az irányítást és a biztonság érdekében megpróbálja lefékezni a vonatot. Ha a tartalék számítógép is hibás, a vonat kézi vezérlésre vált és a vezető feladata lefékezni a vonatot. Előfordulhat azonban, hogy a kézi vezérlő is meghibásodik, ilyenkor nem lehet lefékezni a vonatot és veszélyes helyzet áll elő. Emellett akkor is veszélyes helyzet áll elő, ha a számítógép vagy a vezető megpróbálja ugyan lefékezni a vonatot, de a fékrendszer hibája miatt ez nem sikerül.

A fő számítógép meghibásodásának valószínűsége P_1 , a tartalék számítógépé P_2 , a kézi vezérlőé P_3 a fékrendszeré pedig P_4 . Az események egymástól függetlenek.

- (a) (3 pont) Rajzoljon fel **eseményfát** (event tree) a fő számítógép meghibásodásából (kezdeti esemény) kiindulva!



- (b) (1 pont) Adja meg a veszélyes helyzet kialakulásának valószínűségét!



10. Szolgáltatásbiztonság analízise

Elosztott környezetben egy adat írásának a kérése a következő módon hajtódik végre. A kérés két, redundánsan kapcsolt switch-en keresztül érkezik meg. Utána egy terheléelosztó továbbküldi a három, redundánsan kapcsolt webservert valamelyikének. Ez után ismét egy terheléelosztó küldi tovább a kérést a két, redundánsan kapcsolt adatbázis szerver valamelyikének. A komponensszintű aszimptotikus rendelkezésre állási tényezők (k) a következőképp alakulnak: switch 0,95; terheléelosztó 0,99; webservert 0,9; adatbázis szerver 0,9.

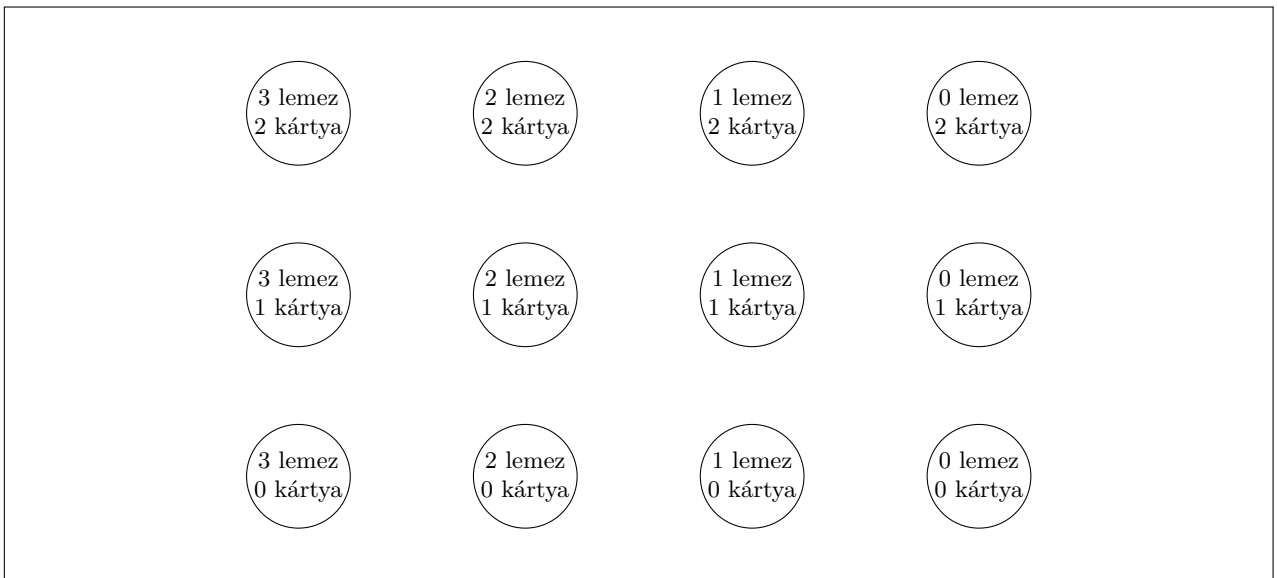
- (a) (3 pont) Rajzoljon fel **megbízhatósági blokkdiagramot** (reliability block diagram) a teljes rendszer rendelkezésre állásáról!

- (b) (1 pont) Adja meg a teljes rendszer aszimptotikus rendelkezésre állási tényezőjét! Egy évben átlagosan hány óra kiesés várható?

11. Szolgáltatásbiztonság analízise

RAID 1 tömbbe kötöttünk 3 merevlemezt (vagyis minden adatot rögzítésre kerül minden lemezre). A lemezek egy RAID vezérlő kártyára vannak kötve, melyből egy hidegtartalék is rendelkezésre áll. Egy merevlemez első meghibásodásának várható ideje (MTFF) 20 000 óra, egy RAID vezérlőé 50 000 óra. Ha az összes lemez meghibásodik vagy már nincs működő vezérlő, a rendszert leállítjuk.

- (a) (3 pont) Egészítse ki a RAID tömb alábbi **folytonos idejű Markov-lánc** (Continuous-Time Markov Chain, CTMC) modelljét az állapotátmenetekkel és átmeneti rátákkal!



- (b) (2 pont) Külön lapra rajzolja fel az állapotok összevonásával kapható, a fentivel ekvivalens minimális állapotszámú folytonos idejű Markov-láncot!