Tesztautomatizálás (EPAM)
1. forduló

Ismertető a feladathoz

Kóstolj bele velünk a teszt automatizáló mindennapi hősök világába! Sokat látott kollégáinkat kértük meg, hogy rakjanak össze nektek egy érdekes feladatsort. Trükkös feladványok, ravasz feleletválasztós tesztek várnak rád, amelyekkel kiderítheted, hogy te is hősből vagy-e! A feladatokat szakmai rutinnal, józan ésszel, csavaros gondolkodással kell megoldanod (vagy akár online kutatással is, nyugi, mi sem tudunk mindent fejből :P). Próbáltuk a mindennapi feladatainkat beleszőni a kérdésekbe, hogy világos képet kapjatok azon sokrétű kihívásokról, amiktől joggal tarthatja magát minden teszt automatizáló szakember mindennapi hősnek!

Ebben a fordulóban, a legtöbbet használt programozási nyelvekkel (JS, Java), reguláris kifejezésekkel és a tesztelés alapjaival kapcsolatos feladatokkal fogsz találkozni.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 30 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 3 pont

Mi a JVM?

- ● A JVM (Java Virtual Machine) java alkalmazások futtatására képes, virtuális működési környezet létrehozásával.
- ○ A JVM egy compiler amely Java programokat tud futtatni.
- ○ JVM a Java Runtime Environment.
- ○ JVM a Java Development Kit, melyet alkalmazások fejlesztésére használnak Java 1.5-től.

Magyarázat a megoldáshoz

2. feladat 0 / 3 pont

Az alábbiak közül melyik(ek) a teszt automatizálás előnye(i) a manuális teszteléssel szemben?

- ○ A teszt futtatás időköltsége nő és a lefedettség is nő.
- ● A teszt futtatás időköltsége csökken, a lefedettség nő.
- ○ A teszt futtatás időköltsége nő, a lefedettség csökken.
- ○ A teszt futtatás időköltsége csökken és a lefedettség is csökken.

Magyarázat a megoldáshoz
Csak a „A teszt futtatás időköltsége csökken, a lefedettség nő.” válasz a helyes. A jól megvalósított teszt automatizálás nagyban lecsökkenti a futtatási időt, miközben a lefedettség is nő.

3. feladat 0 / 3 pont

Mi az eredménye a következő kód utolsó kifejezésének?

```
var x = 1;
var y = true;
var z = false;
x === y ? z == 0 : typeof z;
```

- ○ false
- ○ true
- ● "boolean"
- ○ "number"

Magyarázat a megoldáshoz
Mivel a === operátor szigorú egyenlőség összehasonlítást végez és nem csak az értéket hanem a típust is vizsgálja így az *x===y* hamis lesz, ami a feltételes operátor jobb oldalán lévő elemet adja vissza. A *typeof z* pedig egy *"boolean"*-t ad vissza, mivel a z típusa logikai.

4. feladat 0 / 3 pont

Amikor eredmény jelentési rendszert tervezünk, melyik(ek) a jó megközelítés(ek) gyors módszer biztosítására az olvasónak arra, hogy a teszt futtatás haladásáról kapjon információt?

- ○ Táblázatok
- ● Közlekedési lámpák
- ○ Részletes jelentés, százalékos kiértékeléssel
- ○ Adatbázis az összes eredménnyel

Magyarázat a megoldáshoz
A „Közlekedési lámpák” válasz a helyes, a színkódolás gyors áttekintést ad. Az „Táblázatok” és „Részletes jelentés, százalékos kiértékeléssel” válaszok helytelenek, mert nem támogatják a gyors elemzést. A „Adatbázis az összes eredménnyel” válasz sem helyes, mert a szolgáltatott adatokat előbb még összegezni kellene kiértékelés előtt.

5. feladat 0 / 3 pont

Java programozási nyelvben a final módosító több helyen is használható. Válaszd ki a HAMIS állítást!

- ☐ Egy final osztályt nem lehet örököltetni.
- ☑ Egy final metódust nem lehet túlterhelni vagy elfedni a leszármazott osztályokban.
- ☑ A final változót inicializálni kell deklarálásnál.
- ☐ A final változót csak egyszer lehet inicializálni.

Magyarázat a megoldáshoz
Egy final változót csak egyszer lehet inicializálni, de nem kötelező rögtön a deklarálásnál.

6. feladat 0 / 3 pont

Mikor fogja a Garbage collector kiüríteni egy adott értékhez rendelt memóriát?

- ● Ha nincs több hivatkozás az adott értékre.
- ○ Ha manuálisan elindítok egy garbage collection-t a "triggerGrabageCollection" metódus használatával.
- ○ Soha (Csak ha frissítjük az oldalt.)
- ○ Amint a felhasználó használja a delete operátort.

Magyarázat a megoldáshoz
A garbage collector célja az, hogy figyelje a memória kiosztását és megállapítsa, hogy mikor nincs már szükség egy adott változó/érték lefoglalt memória blokkjára (már nem hivatkozik rá semmi), ekkor fogja kiüríteni azt.

7. feladat 0 / 1 pont

Melyik válasz illeszkedik a következő reguláris kifejezéshez:

```
^[0-9]{3}-[0-9]{2,4}$
```

- ● 213-99
- ○ 98090
- ○ 333-24444
- ○ 1-2345

Magyarázat a megoldáshoz
[0-9]{3} : 3 számjegyű számok illeszkednek (0-9)
- : A "-" karakter illeszkedik rá
[0-9]{2,4} : 2,3 vagy 4 számjegyű számok illeszkednek rá

Vissza a kategóriáimhoz

Tesztautomatizálás (EPAM)
2. forduló

Ismertető a feladathoz

Kóstolj bele velünk a teszt automatizáló mindennapi hősök világába! Sokat látott kollégáinkat kértük meg, hogy rakjanak össze nektek egy érdekes feladatsort. Trükkös feladványok, ravasz feleletválasztós tesztek várnak rád, amelyekkel kiderítheted, hogy te is hősből vagy-e! A feladatokat szakmai rutinnal, józan ésszel, csavaros gondolkodással kell megoldanod (vagy akár online kutatással is, nyugi, mi sem tudunk mindent fejből :P). Próbáltuk a mindennapi feladatainkat beleszőni a kérdésekbe, hogy világos képet kapjatok azon sokrétű kihívásokról, amiktől joggal tarthatja magát minden teszt automatizáló szakember mindennapi hősnek!

Ebben a fordulóban, a legtöbbet használt programozási nyelvekkel (JS, Java), reguláris kifejezésekkel, a HTTP szabvánnyal és a tesztelés alapjaival kapcsolatos feladatokkal fogsz találkozni.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 30 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 3 pont

Kiknek kell visszajelzést és információt adni a teszt automatizálási mérnököknek, ha új rendszerfunkciók miatt kell bővíteniük egy már meglévő teszt automatizálási rendszert?

○ Üzleti elemzők

○ Felsővezetőknek

● Teszt tervezőknek, üzleti tartományi szakértelemmel (domain expertise)

○ Rendszeradminisztrátoroknak

Magyarázat a megoldáshoz

Az üzleti tartományi szakértelemmel rendelkező teszt tervezőknek a feladata, hogy a teszt rendszer helyesen működjön az új funkciókra is. A többi válasz mind helytelen, ezek a szereplők nem vesznek részt a teszt tervezés folyamatában közvetlenül.

2. feladat 0 / 2 pont

Melyik reguláris kifejezéshez illeszkednek a következő szavak: *revolution; revolutionary; revolutionaries*?

○

```
revolution[a-z]?
```

●

```
revolution[a-z]*
```

○

```
revolution[a-z]+
```

Magyarázat a megoldáshoz

- A "*" karakter az előtte lévő [a-z] halmazból 0 vagy több elem is szerepelhet benne, így erre illeszkedik mind a 3 szó.
- A "+" karakter legalább egy karaktert vár a halmazból, így a "revolution" nem illeszkedik.
- A "?" karakter pedig 0 vagy 1 elemet vár a halmazból, így a "revolutionaries" nem illeszkedik.

3. feladat 0 / 1 pont

Az alábbiak közül melyik nem HTTP metódus?

○ POST

● SUBMIT

○ DELETE

○ OPTIONS

Magyarázat a megoldáshoz

4. feladat 0 / 4 pont

Hány szintaktikai hiba van az alábbi, **Jest** tesztelési keretrendszerben írt JavaScript kódban? Válasznak egyetlen egész számot adj meg.

```
describ('my beverage', () = {
  test('is delicious, () => {
    expect(myBeverage.delicious).toBeTruthy();
  });
  it('is not sour', () => {
    expect(myBeverage.sour).toBeFalse();
  });
});
```

A megoldások:
2
4

Magyarázat a megoldáshoz

A szintaktikai hibák a következők:

1. A describe metódus arrow függvényéből hiányzik a > jel.

2. Az első teszteset neve után hiányzik a záró aposztróf.

Nem szintaktikai hibák (nem számítanak bele a helyes válaszba):

1. A describe szó végéről hiányzik az e betű.

2. A második tesztesetnél használt toBeFalse metódus nem létezik, helyette a toBeFalsy metódust lehetne használni.

A helyes kód:

```
describe('my beverage', () => {

 test('is delicious', () => {

  expect(myBeverage.delicious).toBeTruthy();

 });


 it('is not sour', () => {

  expect(myBeverage.sour).toBeFalsy();

 });

});
```

5. feladat 0 / 3 pont

Szeretnénk kiíratni a következő objektum elemeit, ahogy a lenti kimeneten látható.

Melyik ciklussal **nem** lehet ezt megtenni?

```
const tabla = { 1: "a", 2: "b", 3: "c" };
```

Kimenet:

```
1: a
2: b
3: c
```

○

```
for (const elem in tabla) {
    console.log(`${elem}: ${tabla[elem]}`);
};
```

○

```
for (let i = 1; i <= 3; i++) {
    console.log(`${i}: ${tabla[i]}`);
};
```

○

```
let l = 1;
while (l <= Object.keys(tabla).length) {
    console.log(`${l}: ${tabla[l]}`);
    l++;
}
```

●

```
for (const elem of tabla) {
    console.log(`${elem}: ${tabla[elem]}`);
}
```

Magyarázat a megoldáshoz

A *for...of* az adott tömb értékeit járja be és nem használató objektumokra.

6. feladat 0 / 4 pont

Mi lesz a kimenete a következő osztály main metódusának? Válaszodnak a pontos kiírt szöveget add meg (idézőjel nélkül, pl: hello).

```
public class RTExcept {
    public static void throwit() {
        System.out.print("throwit ");
        throw new RuntimeException();
    }

    public static void main(String [] args) {
        try {
            System.out.print("hello ");
            throwit();
        } catch (Exception re ) {
            System.out.print("caught ");
        } finally {
            System.out.print("finally ");
        }
        System.out.println("after");
    }
}
```

A megoldások:
hello throwit caught finally after
hello thorwit caught finally after

Magyarázat a megoldáshoz

A "hello " szöveg kiíródik, mivel ez az első sora a main metódusnak, majd meghívódik a throwit metódus, amely kiírja a "throwit " szót, és kiváltódik egy futásidő kivétel. Ezután a catch block-ban kiíródik a "caught " szöveg, ezután a finally ág írja ki a szöveget, és végül az "after" karakterlánc kerül kiíratásra.

7. feladat 0 / 3 pont

Mi fog történni az osztály futtatásakor?

```
class Battery {
    static int x = 1;
    public static void main(String[] args) throws Throwable {
        try {
            if (x == 1) throw new Throwable();
            System.out.println("try ");
        } catch (Exception e) {
            System.out.println("exc ");
        } finally {
            System.out.println("fin ");
        }
    }
}
```

○ Kiíródik a "try" szöveg.

○ Kiíródik a "exc fin" szöveg.

● Kiíródik a "fin" szöveg, majd az alkalmazás egy runtime exception kivétellel leáll.

○ Kiíródik a "exc fin" szöveg, majd az alkalmazás egy runtime exception kivétellel leáll.

Magyarázat a megoldáshoz

Mivel egy Throwable kivételt dobunk, nem pedig Exceptiont, ezért csak a finally ág fog végrehajtódni a try-catch blokkban.

Vissza a kategóriáimhoz

Tesztautomatizálás (EPAM)
3. forduló

Ismertető a feladathoz

Kóstolj bele velünk a teszt automatizáló mindennapi hősök világába! Sokat látott kollégáinkat kértük meg, hogy rakjanak össze nektek egy érdekes feladatsort. Trükkös feladványok, ravasz feleletválasztós tesztek várnak rád, amelyekkel kiderítheted, hogy te is hősből vagy-e! A feladatokat szakmai rutinnal, józan ésszel, csavaros gondolkodással kell megoldanod (vagy akár online kutatással is, nyugi, mi sem tudunk mindent fejből :P). Próbáltuk a mindennapi feladatainkat beleszőni a kérdésekbe, hogy világos képet kapjatok azon sokrétű kihívásokról, amiktől joggal tarthatja magát minden teszt automatizáló szakember mindennapi hősnek!

Ebben a fordulóban, a két fő teszt automatizálási területtel (UI és API), reguláris kifejezésekkel, a HTTP szabvánnyal és a tesztelés alapjaival kapcsolatos feladatokkal fogsz találkozni, illetve a teszt automatizálás gyakorlati alkalmazásával és metrikáival.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 30 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 3 pont

Az alábbiak közül melyik(ek) a legjobb érv(ek) egy hibajavítás megerősítő tesztjének automatizálására?

○ Befoltozni egy rést a meglévő automatizálási rendszerben.

● Hogy megbizonyosodjunk, a javítás működik most, és a jövőben is fog.

○ Hogy alátámasszuk a hiba megtalálására fordított időt.

○ Hogy ellenőrizzük a konfigurációkezelési folyamatokat.

Magyarázat a megoldáshoz

Arról próbálunk megbizonyosodni, hogy a javítás működik most, és a jövőben is, és nem romlik el újra, például egy esetleges konfigurációváltozás miatt.

A „Befoltozni egy rést a meglévő automatizálási rendszerben.” válasz helytelen, hiába vannak esetleges lyukak a lefedettségben, a megerősítő tesztek automatizálásának nem célja ezeknek a kezelése.

A „Hogy alátámasszuk a hiba megtalálására fordított időt. ” válasz is helytelen, a megtalált hiba súlyossága hivatott, hogy igazolja a megtalálására fordított időt.

Végezetül, a „Hogy ellenőrizzük a konfigurációkezelési folyamatokat. ” válasz is helytelen, mert ugyan ez egy erre is jó mellékhatás, de nem teszteljük a konfiguráció egészét. Azt az egyet biztosítjuk, hogy ez a konkrét hiba nem jön elő újra.

2. feladat 0 / 3 pont

Melyik állítás igaz az alábbiak közül?

● A nemfunkcionális tesztelés automatizálható.

○ A funkcionális tesztelés teljes mértékben automatizálható.

○ A performancia tesztelés nem automatizálható, hiszen az elvárt eredményt nem lehet előre meghatározni.

○ Az automatizált teszteket csak éjszaka szabad futtatni, hogy ne okozzon zavart a tesztelendő alkalmazás működésében.

Magyarázat a megoldáshoz

A nemfunkcionális tesztelés automatizálható. A nemfunkcionális tesztelés egy tesztelési típus, melynek célja a nemfunkcionális aspektusok vizsgálata. Egy jó példa nemfunkcionális tesztre lehetne a következő: Hány felhasználó tud bejelentkezni párhuzamosan az adott alkalmazásba. Ez a terület jellemzően automatizált megoldásokkal tesztlhető.

3. feladat 0 / 2 pont

Melyik válaszok illeszkednek az alábbi reguláris kifejezésre?

```
^\w+$
```

☐ Regex 123

☑ Reg_ex123

☑ regex

☐ Reguláris kifejezés

Magyarázat a megoldáshoz

^ a sor elejét jelzi, a $ pedig a sor végét, a \w pedig bármilyen szóban előforduló karakter helyén állhat (a-zA-Z0-9_), amiből a +-al megkövetelünk legalább 1 karaktert.

4. feladat 0 / 4 pont

Vegyük a következő html struktúrát.

```
<div class="question">
  <span class="bold">What is the answer?</span>
  <label for="answer1">
    <input type="radio" value="1" /> This one
  </label>
  <label for="answer2" class="bold">
    <input type="radio" value="2" /> This other one
  </label>
  <span class="bold">Be wise!</span>
</div>
```

Az alábbi szelektorok közül melyik választja ki csak a második válaszlehetőséget?

● label.bold

○ .bold

○ #question > label.bold

○ .question .bold

Magyarázat a megoldáshoz

A .bold több element szelektál, a .question .bold szintén. A #question > label.bold egyetlen element sem szelektál, mivel egy question id-jú element vár, ami nincs.

5. feladat 0 / 4 pont

Az alábbiak közül melyik státuszkód kiértékelés helyes Postman keretrendszerben?

○

```
pm.response.get.status(200)
```

○

```
pm.response.to.get.status(200)
```

●

```
pm.response.to.have.status(200)
```

○

```
pm.response.assert.status(200)
```

Magyarázat a megoldáshoz

6. feladat 0 / 2 pont

Az alábbi HTTP metódusok közül, melyek a "biztonságosak"? (Azaz konvenció szerint információ szerzésre használandó, és nem módosítja a szerver állapotát.)

☑ GET

☐ PATCH

☑ OPTIONS

☐ PUT

Magyarázat a megoldáshoz

A GET és az OPTIONS metódusok elvben nem kellene hogy megváltoztassák a szerver állapotát, ezek a metódusok a HTTP metódus definíciók szerint biztonságosak. A PATCH és PUT metódusok adat módosítására/hozzáadására szolgálnak, így ezek módosítják a szerver állapotát.

7. feladat 0 / 2 pont

Hány darab tesztesetre lenne minimum szükség, hogy 100%-os utasítás lefedettséget (statement coverage) érjünk el, az alábbi kódrészletben:

```
if (x == 3) {
    System.out.println("FooX");
    if (y == 3) {
        System.out.println("FooY");
    } else {
        System.out.println("BarY");
    }
} else {
    System.out.println("BarX");
}
```

○ 1

○ 4

● 3

○ 5

Magyarázat a megoldáshoz

Az alábbi három teszteset lefedi az összes utasítást:

1. x=3, y=3

2. x=3, y!=3 (e.g. y=0)

3. x!=3, bármilyen y (e.g. y=0)

Vissza a kategóriáimhoz

Tesztautomatizálás (EPAM)
4. forduló

Ismertető a feladathoz

Kóstolj bele velünk a teszt automatizáló mindennapi hősök világába! Sokat látott kollégáinkat kértük meg, hogy rakjanak össze nektek egy érdekes feladatsort. Trükkös feladványok, ravasz feleletválasztós tesztek várnak rád, amelyekkel kiderítheted, hogy te is hősből vagy-e! A feladatokat szakmai rutinnal, józan ésszel, csavaros gondolkodással kell megoldanod (vagy akár online kutatással is, nyugi, mi sem tudunk mindent fejből :P). Próbáltuk a mindennapi feladatainkat beleszőni a kérdésekbe, hogy világos képet kapjatok azon sokrétű kihívásokról, amiktől joggal tarthatja magát minden teszt automatizáló szakember mindennapi hősnek!

Ebben a fordulóban, a két fő teszt automatizálási területtel (UI és API), reguláris kifejezésekkel, a HTTP szabvánnyal és a tesztelés alapjaival kapcsolatos feladatokkal fogsz találkozni, illetve a teszt automatizálás gyakorlati alkalmazásával és metrikáival.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 30 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 3 pont

Tegyük fel, hogy egy olyan rendszert tesztelsz, amihez havonta jönnek szervízcsomagok. Egyszerre több verzió is működik élesben a rendszerből. A meglévő teszt automatizálási megoldásod összetett, és biztosítanod kell, hogy konzisztens marad a különböző környezetek között. Hogyan tudod biztosítani, hogy ugyanazok a tesztek fussanak le mindig, minden környezetre?

○ Mindig kézzel frissíteni a teszt automatizálási megoldást, ha frissül az alkalmazás.
○ Visszaállni csak manuális tesztelésre.
● Egy központi tárból frissen tartani a megoldást, mindegyik éles rendszerre.
○ Kifejleszteni egy eszközt, amivel követni lehet az összes korábbi futás eredményét.

Magyarázat a megoldáshoz

Ha mindig a központi tárból másoljuk le a megoldást, biztosíthatjuk a konzisztenciát.

Az „Mindig kézzel frissíteni a teszt automatizálási megoldást, ha frissül az alkalmazás. ” válasz helytelen, mert nem valósítható meg értelmesen.

A „Visszaállni csak manuális tesztelésre.” válasz is helytelen, a manuális teszteléshez való visszatérés indokolatlan.

Végezetül, a „Kifejleszteni egy eszközt, amivel követni lehet az összes korábbi futás eredményét. ” válasz is helytelen, a korábbi eredmények vizsgálata nincs közvetlen összefüggésben az esetleges konzisztenciahibákkal.

2. feladat 0 / 3 pont

Az alábbi esetek közül melyik javasolható leginkább érdemesnek automatizálásra egy webshop esetében?

○ Termék keresése cikkszám alapján, majd a találatok rendezése népszerűség szerint.
○ A kosár kiürítése 100 elemet meghaladó tételszám esetén.
● Vásárlás véglegesítése és fizetés.
○ Kijelentkezés után felugrik-e a sikeres kijelentkezésről szóló értesítés.

Magyarázat a megoldáshoz

Vásárlás véglegesítése és fizetés. Ugyan mindegyik eset jó jelölt lenne teszt automatizálás szempontjából, mégis az üzleti folyamat szempontjából legkrititikusabb területeket tekintjük magasabb prioritásúnak automatizálási szempontból.

3. feladat 0 / 3 pont

Szeretnék készíteni egy reguláris kifejezést amire csak a "k" karakter illeszkedik, de csak akkor ha előtte a "má" kifejezés áll. Melyik reguláris kifejezést kellene használnom?

●
```
(?<=má)k
```
○
```
má(?=k)
```
○
```
má(k?)
```
○
```
(má?)k
```

Magyarázat a megoldáshoz

A következő reguláris kifejezés a helyes mivel a "lookbehind operatort" használja ("(?<=y)x") ami akkor illeszkedik a fő patternre, ha a prefix előtte szerepel.
A má(?=k) kifejezésre illeszkedik minden "má" kifejezés ha utánna "k" betü áll.
A má(k?) kifejezésre illeszkedik a "má" és a "mák" kifejezés.
A (má?)k kifejezésre pedig minden "k" és "mák" kifejezés is illeszkedik.

4. feladat 0 / 4 pont

Adott az alábbi HTML forráskód:

```
<div class="question">
    <span class="bold">What is your favorite language</span>
    <form>
      <select>
        <option class="lang" value="ruby">Ruby</option>
        <option class="lang" value="c">C</option>
        <option class="lang " value="php">PHP</option>
        <option class="lang" value="js">JavaScript</option>
        <option class="lang star-wars" value="jar">
          Galactic Basic Standard
        </option>
      </select>
    </form>
    <span class="bold">May the force be with you!</span>
</div>
```

Az alábbi szelektorok közül melyik választja ki a programozási nyelveket?

○ option.lang
● option.lang:not(.star-wars)
○ .lang
○ .ruby.c.php.js

Magyarázat a megoldáshoz

Az option.lang és a .lang túl általános, minden opció-ra illeszkedik. A .ruby.c.php.js pedig csak olyan elemenre illeszkedne, ami mind a 4 osztályt tartalmazza, olyan opció pedig nincs, illetve ezek mind értékek az egyes opciókon és nem osztályok.

Az option.lang:not(.start-wars) megtalál minden element amin van lang osztály, de nem rendelkezik a zárójelek között specifikált lokátorral, azaz star-wars osztállyal.

5. feladat 0 / 4 pont

Melyik kifejzéssel értékelnéd ki Postmanben, hogy a válaszban kapott type értéke egyike-e a "Subscriber", "Customer" vagy "User" értékeknek?

○
```
pm.expect(pm.response.json().type)
  .to.be.anyOf(["Subscriber", "Customer", "User"]);
```
○
```
pm.expect(pm.response.json().type)
  .to.be.partOf(["Subscriber", "Customer", "User"]);
```
○
```
pm.expect(pm.response.json().type)
  .to.be.elementOf(["Subscriber", "Customer", "User"]);
```
●
```
pm.expect(pm.response.json().type)
  .to.be.oneOf(["Subscriber", "Customer", "User"]);
```

Magyarázat a megoldáshoz

6. feladat 0 / 2 pont

Az alábbiak közül melyik nem része a SOAP üzenetnek?

○ Header
● Footer
○ Body
○ Fault

Magyarázat a megoldáshoz

7. feladat 0 / 3 pont

Tekintsük az alábbi kódrészletet:

```
public static String fizzBuzz(int number) {
    if (number % 15 == 0) {
        return "fizzbuzz";
    } else if (number % 5 == 0) {
        return "buzz";
    } else if (number % 3 == 0) {
        return "fizz";
    }
    return String.valueOf(number);
}
```

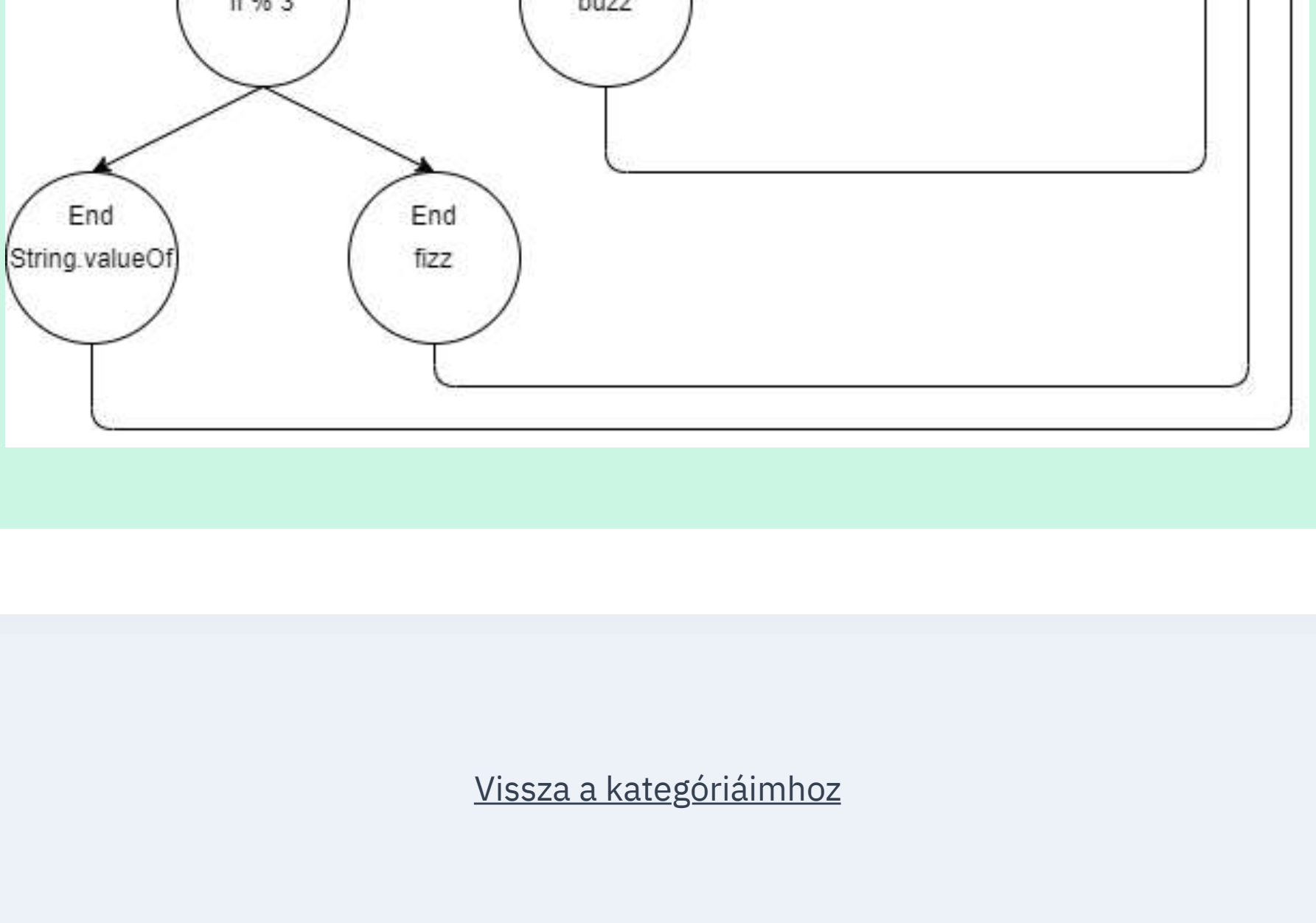
Mekkora a **ciklomatikus komplexitása** a fizzBuzz metódusnak? (válaszként csak egyetlen egy pozitív egész számot adj meg)

A megoldás:
4

Magyarázat a megoldáshoz

Ha felrajzoljuk a kódrészletet mint egy erősen összefüggő gráfot, akkor 10 csomópontot és 7 élt kapunk.

Igy a képlet alapján: 10-7+1=4



Vissza a kategóriáimhoz

Tesztautomatizálás (EPAM)
5. forduló

Ismertető a feladathoz

Kóstolj bele velünk a teszt automatizáló mindennapi hősök világába! Sokat látott kollégáinkat kértük meg, hogy rakjanak össze nektek egy érdekes feladatsort. Trükkös feladványok, ravasz feleletválasztós tesztek várnak rád, amelyekkel kiderítheted, hogy te is hősből vagy-e! A feladatokat szakmai rutinnal, józan ésszel, csavaros gondolkodással kell megoldanod (vagy akár online kutatással is, nyugi, mi sem tudunk mindent fejből :P). Próbáltuk a mindennapi feladatainkat beleszőni a kérdésekbe, hogy világos képet kapjatok azon sokrétű kihívásokról, amiktől joggal tarthatja magát minden teszt automatizáló szakember mindennapi hősnek!

Ebben a fordulóban, a két fő teszt automatizálási területtel (UI és API), követelményekkel és reguláris kifejezésekkel kapcsolatos feladatokkal fogsz találkozni, illetve a teszt automatizálás gyakorlati alkalmazásával.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 35 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 10 pont

Melyik állításban szereplő karakterisztikák határoznak meg egy jól elkészített követelményt?

○ Megfelelő méretű, Tesztelhető, Visszavonható

○ Visszavonható, Változtatható, Független

● Független, Jól becsülhető, Tesztelhető

○ Tesztelhető, Értéket hordozó, Könnyen fejleszthető

Magyarázat a megoldáshoz

Az **INVEST** technika segít a megfelelő karakterisztikákkal bíró követelményeket elkészíteni. A betűk feloldása a következő:

- **I - Independent -** Független a többi követelménytől
- **N - Negotiable -** A követelmény tartalma módosítható, amíg a fejlesztése el nem kezdődik
- **V - Valuable -** A követelménynek tartalmaznia kell valamilyen értéket/új funkcionalitást, ami a végfelhasználónak pluszt jelent
- **E - Estimable -** A követelménynek jól becsülhetőnek kell lennie, hogy lehessen tudni, hogy mennyi erőforrásba kerülni majd az elkészítése, illetve mikor kerülhet szállításra az új funkcionalitás
- **S - Sized Appropriately -** Egy követelménynek megfelelő méretűnek kell lennie, tartalomtól függően nem szabad sem túl kicsire vagy túl nagyra tervezni
- **T - Testable -** A követelménynek könnyen tesztelhetőnek kell lennie, nem maradhatnak nyitott kérdések, illetve konkrét számokkal kell konkretizálni egy új funkció működését (pl.: A legördülő listában 5 elemnek kell lennie)

2. feladat 0 / 8 pont

Az alábbiak közül, melyik válasz indokolja leginkább, hogy egy szoftverhiba javításához annak helyességét megerősítő automata tesztet készítsünk? Azért, hogy...

○ pótoljuk a meglévő automata tesztek hiányosságait.

● meggyőződjünk róla, hogy a javítás működik és a jövőben is működni fog.

○ igazoljuk a hiba megtalálására eltöltött időt.

○ teszteljük a az automata tesztek konfigurációs lehetőségeit.

Magyarázat a megoldáshoz

Fontos, hogy az alkalmazás jövőbeli verziói esetén is megvizsgáljuk a javítást. Illetve egy meglévő teszt segíthet ellenőrizni a javítás helyességét is.

3. feladat 0 / 6 pont

Melyik válasz illeszkedik az alábbi reguláris kifejezésre:

```
^A szoftvertesztelés (hasznos|(?!rossz)|szükséges) (webapplikációknál|vi
```

● A szoftvertesztelés hasznos webapplikációknál szerintem

○ A szoftvertesztelés rossz webapplikációknál szerintem

○ A szoftvertesztelés szükséges videójátékoknál az oktatóm szerint

○ A szoftvertesztelés hasznos videójátékoknál

Magyarázat a megoldáshoz

^ a sor elejét jelzi, a () pedig csoportokat hoznak létre "vagy" kapcsolattal, kivéve a "rossz" szóra, mivel ott egy "Negative Lookahead" van használva, ezért a "rossz" szót nem lehet használni. A végén min 1 max 10 kisbetűs, ékezetek nélküli karaktert és space-t lehet használni a [a-z]{1,10} miatt. A $ a sor végét jelöli.

4. feladat 0 / 10 pont

Elbukna-e a következő **Selenium** alapú **Cucumber** teszt az alábbi UI elemre vonatkozóan?

```
<button class="ok-button curly-button"
    type="button" aria-label="OK"/>
```

```
Then('the OK button should be visible', async () => {
    const $okButton = driver.findElement(
        by.css(".ok-button:not(.curly)")
    );
    expect(await $okButton.isDisplayed()).to.be.true;
});
```

○ Igen, mert a Cucumber teszt végrehajtó keretrendszer nem képes aszinkron műveletek kezelésére.

● Nem, mert az elem lokalizálható a megadott lokátorral és nem rendelkezik olyan attribútummal mely alapján ne lenne megjelenítve.

○ Igen, mert az adott elem class attribútuma nem tartalmazhatja a curly stringet.

○ Nem, mert a hibásan megadott lokátor miatt az alapbeállításként megadott "PASSED" állapottal térne vissza.

Magyarázat a megoldáshoz

Nem fog elbukni a teszt, hiszen az elem lokalizálható (nem rendelkezik .curly class attribútummal) és nem rendelkezik a láthatóságát befolyásoló attribútumokkal (pl display=none).

5. feladat 0 / 10 pont

Az alábbi linken elérhető dokumentum tartalmazza egy API egyik metódusának leírását, valamint a metódus tesztelésére néhány tesztesetet: https://epa.ms/oitm-my-cool-library

Válaszd ki, hogy ezek közül melyek a **valid negatív tesztesetek**.

☐ A

☑ B

☐ C

☑ D

Magyarázat a megoldáshoz

A B és a D opciók a negatív teszt esetek, mivel ezekben azt teszteljük, hogy a service metódus megfelelően le tudja-e kezelni a lehetséges hibákat. Ellenben az A és C pedig pozitív teszt esetek.

Vissza a kategóriáimhoz

Tesztautomatizálás (EPAM)
6. forduló

Ismertető a feladathoz

Kóstolj bele velünk a teszt automatizáló mindennapi hősök világába! Sokat látott kollégáinkat kértük meg, hogy rakjanak össze nektek egy érdekes feladatsort. Trükkös feladványok, ravasz feleletválasztós tesztek várnak rád, amelyekkel kiderítheted, hogy te is hősből vagy-e! A feladatokat szakmai rutinnal, józan ésszel, csavaros gondolkodással kell megoldanod (vagy akár online kutatással is, nyugi, mi sem tudunk mindent fejből :P). Próbáltuk a mindennapi feladatainkat beleszőni a kérdésekbe, hogy világos képet kapjatok azon sokrétű kihívásokról, amiktől joggal tarthatja magát minden teszt automatizáló szakember mindennapi hősnek!

Ebben a fordulóban mindennapi, a JS és Java teszt automatizálás területéről vett és a folyamatos integrációhoz kapcsolódó feladatokkal fogsz találkozni.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 45 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 13 pont

Az alábbiak közül melyik Continous Integration és nem Deployment vagy Delivery ciklus?

○ CodeChange -> Build -> Test -> PushChange -> Release -> Deploy

● CodeChange -> Build -> Test -> PushChange

○ CodeChange -> Build -> Test -> PushChange -> Release

Magyarázat a megoldáshoz

2. feladat 0 / 16 pont

Az alábbi függvény segítségével egy tömb páros elemeihez hozzáadunk 10-et és a páratlanokat változatlanul hagyjuk, majd visszaadjuk az új kapott tömböt. Melyik beépített Javascript függvény neve hiányzik az aláhúzott részen, hogy a fentiek teljesüljenek?

```
const elsoTomb = [2, 3, 10, 4, 3, 7, 8, 10];
function tombModositas(tomb) {
    return tomb.______(elem => {
        if (elem % 2 == 0) {
            elem += 10;
        }
        return elem;
    });
}
console.log(tombModosit(elsoTomb));
// [ 12, 3, 20, 14, 3, 7, 18, 20 ]
```

○ filter

● map

○ forEach

○ shift

Magyarázat a megoldáshoz

A map függvény a helyes válasz mert: A map() metódus egy új tömböt hoz létre, amely a hívó tömb minden elemén egy megadott függvény meghívásának eredményeivel kerül feltöltésre.

A filter új tömböt hoz létre az összes olyan elemmel, amely megfelel a megadott függvény által végrehajtott tesztnek.

A forEach az adott tömbön operál amire meghívjuk és undefined a visszatérési értéke.

A shift kitörli az első elemet a tömbből és azzal tér vissza.

3. feladat 0 / 16 pont

Tekintsük az alábbi Cucumber feature file snippet-et:

```
Background:
  Given the home page is opened
  And the Regisztráció header button is clicked
  And the Cookie disclaimer is closed
```

Milyen annotációt lehet a step definition metódusra rakni a következő kódban, a ______ helyre, hogy lefedjük a "Given the home page is opened" sort?

```
______
public void theHomePageIsOpened() {
  clearBrowserCookies();
  homePage = open("/", SpotifyHomePage.class);
}
```

☑ `@Given("the home page is opened")`

☑ `@When("the home page is opened")`

☐ `@Given("Given the home page is opened")`

☐ `@Given("The home page is opened")`

Magyarázat a megoldáshoz

@Given("the home page is opened"), a zárójelekben a feature fileban a Given utáni rész jön, az annotáció maga pedig az első szó. Viszont a Cucumber nem különbözteti meg a két annotációt, és a @Given("the home page is opened") és a @When("the home page is opened") ugyan úgy használható.

Vissza a kategóriáimhoz

Tesztautomatizálás (EPAM)
7. forduló

Ismertető a feladathoz

Kóstolj bele velünk a teszt automatizáló mindennapi hősök világába! Sokat látott kollégáinkat kértük meg, hogy rakjanak össze nektek egy érdekes feladatsort. Trükkös feladványok, ravasz feleletválasztós tesztek várnak rád, amelyekkel kiderítheted, hogy te is hősből vagy-e! A feladatokat szakmai rutinnal, józan ésszel, csavaros gondolkodással kell megoldanod (vagy akár online kutatással is, nyugi, mi sem tudunk mindent fejből :P). Próbáltuk a mindennapi feladatainkat beleszőni a kérdésekbe, hogy világos képet kapjatok azon sokrétű kihívásokról, amiktől joggal tarthatja magát minden teszt automatizáló szakember mindennapi hősnek!

Ebben a fordulóban mindennapi, a JS és Java teszt automatizálás területéről vett, és a folyamatos integrációhoz kapcsolódó feladatokkal fogsz találkozni.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 45 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 13 pont

Continuous Delivery/Deployment ciklus esetén melyik process biztosítja a legmagasabb Quality Assurance-t az alábbiak közül?

○ A tervezett release major és minor verziójával megegyező (szemantikus verziózás szerint) legújabb verziójú artifact promótálása és deployolása.

○ A (GitFlow branching stratégia szerinti) release branch aktuális állapotából célkörnyezetre optimalizált artifact építése és deployolása.

○ A már teljes QA processzen átesett forráskódból production környezetre optimalizált artifact építése és deployolása.

● A már teljes QA processzen átesett artifact verzió promótálása és deployolása.

Magyarázat a megoldáshoz

2. feladat 0 / 20 pont

Tegyük fel, hogy UI teszteket kell írnod az Országos IT Megmérettetés honlapjának (https://ithon.info/megmerettetes) teszteléséhez. Az egyik tesztesetben az oldal első szintű fejlécének (h1) szövegét szeretnéd ellenőrizni (ami: IV. Országos IT Megmérettetés). A böngésző automatizáláshoz és a UI elemekkel való interakcióhoz a Puppeteer keretrendszert kell használnod. Az alábbi tesztesetek közül melyik bukik el?

☐

```
const text = await  page.$eval("h1", element => element.textContent);
expect(text).toContain("Országos IT Megmérettetés 2020");
```

☑

```
const heading1 = await page.$("h1");
const text = await heading1.getProperty("textContent");
expect(text).toContain("Országos IT Megmérettetés 2020");
```

☑

```
const heading1 = await page.$("h1");
const text = await  heading1.evaluate(node => node.innerText);
expect(text).toContain("ORSZÁGOS IT MEGMÉRETTETÉS 2020");
```

☐

```
const heading1 = await page.$("h1");
const text = await  page.evaluate(
    element => element.textContent, heading1
);
expect(text).toContain("Országos IT Megmérettetés 2020");
```

Magyarázat a megoldáshoz

A getProperty() metódus ebben az esetben JSHandle objektumot ad vissza, nem pedig a tényleges szöveget. Ahhoz, hogy megkapjuk az elem szövegét, még meg kell hívni rá a jsonValue() metódust, így:

const text = await (await heading1.getProperty("textContent")).jsonValue();

3. feladat 0 / 20 pont

Tekintsük az alábbi osztályokat:

```
public class AccountService {
    private UserService userService = new UserService();
    //...
    public void activateAccount(String username) {
        userService.activateAccount(username);
        //...
    }
    //...
}

public class UserService {
    //...
    public void activateAccount(String username) {
        System.out.println("username: " + username + " activated");
    }
    //...
}
```

Tegyük fel, hogy le szeretnéd tesztelni az AccountService class activateAccount metódusát, viszont ehhez a UserService osztály activateAccount metódusát le szeretnéd cserélni egy mock metódusra, ami úgy működik ahogy neked szükséges. A teszteléshez a **PowerMock+JUnit4** frameworkoket használhatod.

Mi szükséges ehhez írnod a ____ részre?

```
@RunWith(PowerMockRunner.class)
@PrepareForTest({AccountService.class})
public class AccountServiceTest {
    private AccountService accountService;

    @Test
    public void testAuch() throws Exception {
        ____
        accountService.activateAccount("myUserName");
        //assertions…..
    }
}
```

☐ Ezt lehetetlen megcsinálni PowerMockito segítségével.

☑

```
UserService mockUserService = PowerMockito.mock(UserService.class);
PowerMockito.whenNew(UserService.class)
    .withNoArguments()
    .thenReturn(mockUserService);
accountService = new AccountService();
PowerMockito.doAnswer(c -> {
        //mocked logic
        return "";
    })
    .when(mockUserService)
    .activateAccount("testData");
```

☐

```
UserService mockUserService = Mockito.mock(UserService.class);
mockUserService.when(mockUserService)
    .activateAccount("myUserName")
    .thenReturn(/* ownMockedReturn */);
```

☑

```
PowerMockito.whenNew(UserService.class)
    .withNoArguments()
    .thenReturn(mockUserService);
accountService = new AccountService();
```

Magyarázat a megoldáshoz

Lehetséges megcsinálni PowerMockito segítségével. A válaszban az első kódsor létrehozza a mock osztályt, a második kódsor pedig felülírja az osztály példányosítását. Miután ezt felülírtuk, létrehozunk egy példányt a tesztelendő osztályból, ebben már a mockolt UserService lesz. Majd felülirjuk a UserService activateAccount metódusát.

Vissza a kategóriáimhoz