

TESZTAUTOMATIZÁLÁS

2. forduló



A kategória támogatója: EPAM

Ismertető a feladathoz

Útmutató:

- A **radio button-os kérdésekre** egy helyes válasz van.
- Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- Badge-eket** a 4.forduló után kapsz majd először.
- Az **adatbekérős kérdéseknél** igyekeztünk minden variációt megadni (kisbetű, nagybetű, szóköz), de ha mégis eltérést tapasztalsz a megoldásokban, kérjük, jelezd felénk!

+1: Azért szólunk, hogy senkit ne a végén érjen meglepetés: a játék nem tipp-mix és csapatkategória sincs! Természetesen akinek nem inge...

Jó versenyzést kívánunk!

Felhasznált idő: 00:00/30:00

Elért pontszám: 0/65

1. feladat 0/5 pont

Melyik hibaüzenetet küldi a szerver az alábbiak közül, amikor átjáróként dolgozik és másik kiszolgálótól érvénytelen választ kapott?

Válasz

- ☒ 502 Bad Gateway
Ez a válasz helyes, de nem jelölted meg.
- ☐ 503 Service Unavailable
- ☐ 504 Gateway Timeout
- ☐ 500 Internal Server Error

Magyarázat

Az 502 Bad Gateway a helyes válasz, akkor küldi a szerver, amikor hibás - invalid választ kap egy másik kiszolgálótól.

A többi felsorolt hibakód jelentése a következő:

503 Service Unavailable: nem elérhető a szolgáltatás, amit a kliens használni szeretne.

504 Gateway Timeout: nem kap választ a szerver a másik kiszolgálótól meghatározott időben.

500 Internal Server Error: a szerver olyan hibát észlel, amire nincs felkészítve így nem tudja, hogyan oldja meg.

2. feladat 0/10 pont

Egy térképkalkulációt fejlesztők készíteni szeretnének egy ellenőrző algoritmust, ami ellenőrzi a bejövő adatokat, hogy megfelelnek-e a követelményeknek. Az alkalmazás szélességi és hosszúsági koordinátákat kap bemenetként az alábbi formátumban:

(szélesség, hosszúság)

(-90.00000, +180.0000)

-90<=Szélesség<=+90 és -180<=Hosszúság<=180.

példák: (+90.0, -147.45); (77.11112223331, 149.99999999); (90, 180)

Érkezhet előjelek nélkül is a bemenet, ekkor pozitívnak és validnak veszi a koordinátákat az alkalmazás. Tizedes jegyek nélkül is érkezhet a bemenet, ez is jó bemenetnek számít.

Az alkalmazás csak ezzel a formátummal tud dolgozni, ezért szükséges hozzá egy szűrő reguláris kifejezés, ami csak a megfelelő formátumú koordinátákat enged át.

A következő reguláris kifejezések közül melyek szűrik ki a rossz bementeket?

Válaszok

☒ `^\\([+-]?90(\\.0+)?|[1-8]\\d(\\.\\d+)?|\\d(\\.\\d+)?),[+-]?{180(\\.0+)?|1[0-7]\\d(\\.\\d+)?|[1-9][0-9](\\.\\d+)?|\\d(\\.\\d+)?}\\)$`
Ez a válasz helyes, de nem jelölted meg.

☐ `^\\((\\+|\\-)?(\\d+)(\\.\\d+)?),((\\+|\\-)?(\\d+|1[0-7][0-9])(\\.\\d+)?))\\)$`

☒ `^\\(((\\+|\\-)?([0-9]|[1-8][0-9])(\\.\\d+)?|90(\\.0+)?)),((\\+|\\-)?([0-9]|[1-9][0-9]|1[0-7][0-9])(\\.\\d+)?|180(\\.0+)?))\\)$`
Ez a válasz helyes, de nem jelölted meg.

☐ `^\\([+-]?(\\d+),[+-]?(\\d+))\\)$`

Magyarázat

`^\\((\\+|\\-)?(\\d+)(\\.\\d+)?),((\\+|\\-)?(\\d+|1[0-7][0-9])(\\.\\d+)?))\\)$` reguláris kifejezés illeszkedik a -90 kisebb és +90 nagyobb szélességi és -180-tól kisebb és +180-tól nagyobb hosszúsági koordinátákra is, ezért ez a válasz helytelen.

`^\\([+-]?(\\d+),[+-]?(\\d+))\\)$` reguláris kifejezés nem illeszkedik a tizedes jegyeket tartalmazó koordinátákra, ezért ez a válasz helytelen.

`^\\([+-]?90(\\.0+)?|[1-8]\\d(\\.\\d+)?|\\d(\\.\\d+)?),[+-]?{180(\\.0+)?|1[0-7]\\d(\\.\\d+)?|[1-9][0-9](\\.\\d+)?|\\d(\\.\\d+)?}\\)$`

és

`^\\(((\\+|\\-)?([0-9]|[1-8][0-9])(\\.\\d+)?|90(\\.0+)?)),((\\+|\\-)?([0-9]|[1-9][0-9]|1[0-7][0-9])(\\.\\d+)?|180(\\.0+)?))\\)$` reguláris kifejezések teljesítik a követelményeket és megszűrik nekünk a jó bementeket, ezért ez a két válasz a helyes.

3. feladat 0/10 pont

Mi lesz a következő kódrészlet kimenete?

```
class Test
{
    int a = 1;
    int b = 2;
```

```

Test func(Test obj)
{
    Test obj3 = new Test();
    obj3 = obj;
    obj3.a = obj.a++ + ++obj.b;
    obj3.b = obj.b;
    return obj3;
}

public static void main(String[] args)
{
    Test obj1 = new Test();
    Test obj2 = obj1.func(obj1);

    System.out.println("obj1.a = " + obj1.a + "   obj1.b = " + obj1.b);
    System.out.println("obj2.a = " + obj2.a + "   obj2.b = " + obj2.b);

}
}

```

Válasz

- ☐ obj1.a = 1 obj1.b = 2
obj2.a = 4 obj2.b = 3
- ☒ obj1.a = 4 obj1.b = 3
obj2.a = 4 obj2.b = 3
Ez a válasz helyes, de nem jelölted meg.
- ☐ Compile error / Fordítási hiba
- ☐ obj1.a = 3 obj1.b = 3
obj2.a = 3 obj2.b = 3

Magyarázat

Az 'obj1' és az 'obj2' ugyanaz az objektum, mivel a func metódusban az 'obj3' az 'obj1' referenciát használja.

4. feladat 0/10 pont

A következő módszerek közül mely(ek) **NEM** a fehér doboz technikák csoportjába tartozik(/nak)?

Válaszok

- ☐ Utasítások lefedettségének vizsgálata
- ☒ Struktúra alapú tesztelés
Ez a válasz helyes, de nem jelölted meg.
- ☒ Hibasejtés
Ez a válasz helyes, de nem jelölted meg.
- ☐ Döntési eredmények meghívásának százalékos végrehajtási arányának vizsgálata

Magyarázat

Utasítások lefedettségének vizsgálata: valós fehér doboz technika. Struktúra alapú tesztelés: a fehér doboz technika másik elnevezése, nem fehér doboz technika. Hibasejtés: tapasztalat alapú technika, nem fehér doboz technika. Döntési eredmények meghívásának százalékos végrehajtási arányának vizsgálata: röviden döntési lefedettség, mely egy valós fehér doboz technika.

5. feladat 0/15 pont

Az alábbiak közül melyik lehet egy működőképes Postman script segítségével készült teszt?

Válasz



```
pm.test("Status test", function () {  
    pm.response.to.have.status(418) ;  
});
```

Ez a válasz helyes, de nem jelölted meg.



```
test('should create a bug report', async ({ request }) => {  
    const newIssue = await request.post(`/repos/${USER}/${REPO}/issues`, {  
        data: {  
            title: '[Bug] report 1',  
            body: 'Bug description',  
        }  
    });  
    expect(newIssue.ok()).toBeTruthy();  
    const issues = await request.get(`/repos/${USER}/${REPO}/issues`);  
    expect(issues.ok()).toBeTruthy();  
    expect(await issues.json()).toContainEqual(expect.objectContaining({  
        title: '[Bug] report 1',  
        body: 'Bug description'  
    }));  
});
```



```
it('adds a todo', () => {  
    cy.request({  
        url: '/todos',  
        method: 'POST',  
        body: {  
            title: 'Write REST API',  
        },  
    })  
    .its('body')  
    .should('deep.contain', {  
        title: 'Write REST API',  
        completed: false,  
    })  
});
```



```
it('should get random male users', async () => {  
    await spec()  
        .get('https://randomuser.me/api')  
        .withQueryParams('gender', 'male')  
        .expectStatus(200);  
});
```

Magyarázat

A

```
pm.test("Status test", function () {  
    pm.response.to.have.status(418) ;  
});
```

kódrészlet a helyes megoldás. A többi példa az alábbi test automation framework-ök példakódja: Playwright, Cypress, PactumJS.

További információ az alábbi linkeken található:

<https://docs.cypress.io/>

<https://playwright.dev/docs/test-api-testing>

<https://pactumjs.github.io/guides/api-testing.html>

<https://learning.postman.com/docs/writing-scripts/test-scripts/>

6. feladat 0/15 pont

Tegyük fel, hogy a megoldásokban szereplő tesztet párhuzamosan szeretnénk lefuttatni pytest és pytest-xdist segítségével, az alábbi utasítással:

pytest -n 2

Az alábbi esetek közül melyek esetében kapunk 10 PASSED eredményt?

Válaszok



```
import pytest  
import string  
from random import choices  
  
def foo(s):  
    return int(s, base=16)  
  
@pytest.mark.parametrize("s", map(lambda s: pytest.param(s, id='choice'), choices(string.hexdigits, k=10)))  
def test_foo(s: string):  
    assert foo(s) == int(s, base=16)
```

◀ Ez a válasz helyes, de nem jelölted meg. ▶



```
import pytest  
import string  
from random import choices  
  
def foo(s):  
    return int(s)  
  
@pytest.mark.parametrize("s", map(lambda s: pytest.param(f"0x{s}", id='choice'), choices(string.hexdigit
```

```
def test_foo(s: string):
    assert foo(s) == int(s)
```

```
import pytest
import string
from random import choices

def foo(s):
    return int(s, base=16)

@pytest.mark.parametrize("s", map(lambda s: pytest.param(s, id=s), choices(string.hexdigits, k=10)))
def test_foo(s: string):
    assert foo(s) == int(s, base=16)
```

```
import pytest
import string
from random import choices

def foo(s):
    return int(s, base=16)

@pytest.mark.parametrize("s", map(lambda s: pytest.param(f"0x{s}", id="0x{s}"), choices(string.hexdigits, k=10)))
def test_foo(s: string):
    assert foo(s) == int(s, base=16)
```

Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Párhuzamos teszt futtatás esetén a pytest subprocessekben fut, és minden subprocess elkészíti a saját tesztgyűjteményét a tesztek neve alapján. Parametrizált tesztek esetében a tesztek neve kiegészül a test id-val, ami random érték esetében random neveket fog eredményezni. Ennek következtében ebben a konkrét esetben a két subprocess két különböző gyűjteményt állít elő, amelyeket a pytest nem tud megfelelően ütemezni, ezért "Different tests were collected between gw0 and gw1." hibával leáll még azelőtt, hogy egyetlen tesztet is lefuttatna.

Ha a `pytest.param`mal átadott paraméterekben azok id-ját statikusra állítjuk ("choice", illetve "0x{s}"), akkor a tesztek neve nem fog random értéket tartalmazni, így a pytest képes lefuttatni őket.

Ha a `pytest.param`mal a random értéket adjuk át az id paraméterben (`id=s`), akkor a pytest nem tudja lefuttatni a tesztek, mivel az teljes teszt név tartalmazni fogja ezeket a random id-kat.

Végezetül, a `string->int` konverzió esetén, ha az nem 10-es számrendszer alapján történik, akkor a `base` paraméterben meg kell adni a számrendszert, így hexadecimális szám esetén a 16-ot még akkor is, ha maga a `string` `0x` prefixet tartalmaz. Így az a megoldás, amiből ez hiányzik, nem jó.



