Anadolu Üniversitesi — Bilişim Teknolojileri MYO — Oyun Geliştirme ve Programlama Programı

OGP105 Oyun Geliştirme Final Sınavı — 7 Ocak 2026

Ad Soyad:	**Öğrenci No:**	**İmza:**

⚠️ **ÖNEMLİ:** Sınav süresi **90 dakikadır**. Soru 1-5 zorunludur. Soru 6 **VEYA** Soru 7+8+9 birlikte cevaplanacaktır. Her iki seçeneğin de puanı 40'tır.

Soru 1

10 puan

		3	
		1	
			1
3		**2**	

Başlangıç tahtası

Bu sınavda **4×4 Mini Sudoku** üzerinde çalışacağız. Tahta 4×4 hücreden oluşur, bu puzzle'ı çözmek için her hücreye 1-4 arası rakamlar yerleştirilir ve her satır/sütunda tüm rakamlar birer kez bulunmalıdır. Boş hücreler `0` ile temsil edilir.

Tahtayı `board` dizisinde saklıyoruz:

```
let board = [];
for (let satir = 0; satir < 4; satir++) {
    board[satir] = [];
    for (let sutun = 0; sutun < 4; sutun++) {
        board[satir][sutun] = 0;
    }
}
```

İstenen: Yukarıdaki kodun altına hangi satırları eklersek `board` dizisi tam olarak soldaki tahtayı temsil eder?

Cevap:

Soru 2a

10 puan

Sudoku oyununda bir satırın geçerli olup olmadığını kontrol eden `isValidRow(row)` fonksiyonunu yazınız. Bu fonksiyon parametre olarak 4 elemanlı bir dizi alır. Satırın geçerli olması için dizide 1, 2, 3 ve 4 rakamlarının her biri **tam olarak bir kez** bulunmalıdır. Geçerliyse `true`, değilse `false` döndürür.

Örnek çağrılar:

```
isValidRow([1, 2, 3, 4])  →  true
isValidRow([1, 3, 4, 3])  →  false
isValidRow([1, 2, 3, 0])  →  false
```

Cevap:

Soru 2b

10 puan

Bir arkadaşınız `isValidRow` fonksiyonunu aşağıdaki gibi yazmıştır:

```
function isValidRow(row) {
    let toplam = row[0] + row[1] + row[2] + row[3];
    return (toplam == 10);  // 1+2+3+4 = 10
}
```

Bu implementasyon neden **her zaman doğru sonuç üretmez**? Sözel olarak açıklayınız ve gerekirse örnek vererek destekleyiniz.

Cevap:

Soru 3

10 puan

Sudoku oyununda tahtanın tamamen dolup dolmadığını kontrol eden `isBoardFull(b)` fonksiyonunu yazınız. Bu fonksiyon parametre olarak 4×4'lük iki boyutlu bir dizi (tahta) alır. Tahtada hiç boş hücre (0 değeri) kalmamışsa `true`, en az bir boş hücre varsa `false` döndürür.

Örnek çağrılar:

```
let tahta1 = [[1,2,3,4], [3,4,1,2], [2,1,4,3], [4,3,2,1]];
isBoardFull(tahta1)  →  true

let tahta2 = [[1,2,3,4], [3,4,1,2], [2,1,0,3], [4,3,2,1]];
isBoardFull(tahta2)  →  false
```

Cevap:

OGP105 Oyun Geliştirme Final Sınavı — 7 Ocak 2026

Ad Soyad:	**Öğrenci No:**	**İmza:**

Soru 4

10 puan

`isValidBoard(b)` fonksiyonu, Sudoku tahtasının tamamının geçerli olup olmadığını kontrol eder. Geçerli bir tahtada **her satır** ve **her sütun** 1, 2, 3, 4 rakamlarını tam olarak birer kez içermelidir. Fonksiyon, daha önce yazdığınız `isValidRow()` fonksiyonunu kullanmaktadır.

Örnek çağrılar:

```
let gecerliTahta = [[1,2,3,4], [3,4,1,2], [2,1,4,3], [4,3,2,1]];
isValidBoard(gecerliTahta)  →  true

let gecersizTahta1 = [[1,2,3,4], [1,2,3,4], [1,2,3,4], [1,2,3,4]];
isValidBoard(gecersizTahta1)  →  false  // satırlar geçerli ama sütunlar değil

let gecersizTahta2 = [[1,2,3,4], [3,4,1,2], [2,1,4,3], [4,3,2,2]];
isValidBoard(gecersizTahta2)  →  false  // son satırda 1 yok, 2 iki kez var
```

Aşağıdaki kod `isValidBoard` fonksiyonunu implemente etmeye çalışıyor ancak **yanlış sonuç döndürüyor**:

```
1  function isValidBoard(b) {
2      for (let satir = 0; satir < 4; satir++) {
3          if (!isValidRow(b[satir])) {
4              return true;
5          }
6      }
7
8      for (let sutun = 0; sutun < 4; sutun++) {
9          let dikeyÇizgi = [];
10         for (let satir = 0; satir < 4; satir++) {
11             dikeyÇizgi[satir] = b[satir][sutun];
12         }
13         if (!isValidRow(dikeyÇizgi)) {
14             return true;
15         }
16     }
17     return false;
18 }
```

İstenen: Bu kodun doğru çalışması için **3 satırda** değişiklik yapınız. Değiştirmeniz gereken satır numaralarını ve bu satırların yeni hallerini yazınız.

Cevap:

Soru 5

10 puan

Aşağıdaki kod, 4×4 Sudoku tahtasını çizen bir p5.js programıdır:

```
let board = [[0,0,3,0], [0,0,1,0], [0,0,0,1], [3,0,2,0]];

function setup() {
    createCanvas(400, 400);
    textAlign(CENTER, CENTER);
}

function draw() {
    for (let satir = 0; satir < 4; satir++) {
        for (let sutun = 0; sutun < 4; sutun++) {
            rect(sutun * 100, satir * 100, 100, 100);
            if (board[satir][sutun] != 0) {
                text(board[satir][sutun], sutun * 100 + 50, satir * 100 + 50);
            }
        }
    }
}
```

Bu oyunda kullanıcı bir hücreye tıkladığında, tıklanan noktanın hangi satır ve sütuna denk geldiğini bulmamız gerekiyor. `getSatirSutun(x, y)` fonksiyonu, koordinatları (x, y) alır ve **[satir, sutun]** şeklinde bir dizi döndürür.

Örnek çağrılar ve beklenen sonuçlar:

```
getSatirSutun(50, 50)          →  [0, 0]  // sol üst hücre
getSatirSutun(350, 150)        →  [1, 3]  // 2. satır, 4. sütun
getSatirSutun(mouseX, mouseY)  →  [3, 2]  // imleç (220, 310) konumundayken
```

İstenen: `getSatirSutun(x, y)` fonksiyonunu yazınız. *(Not: Fonksiyon içinde `mouseX` ve `mouseY` özel değişkenlerini kullanmayınız, parametre olarak gelen x ve y değerlerini kullanınız.)*

Cevap:

Ad Soyad:	**Öğrenci No:**	**İmza:**

Soru 6 (SEÇMELİ - A)

40 puan

Aşağıdaki kod, Sudoku tahtasını ve altındaki sayı seçme alanını çizmektedir:

```
let board = [[0,0,3,0], [0,0,1,0], [0,0,0,1], [3,0,2,0]];

function setup() {
    createCanvas(400, 600);
    textAlign(CENTER, CENTER);
}

function printBoard(b) {
    for (let satir = 0; satir < 4; satir++) {
        for (let sutun = 0; sutun < 4; sutun++) {
            rect(sutun * 100, satir * 100, 100, 100);
            if (b[satir][sutun] != 0) {
                text(b[satir][sutun], sutun * 100 + 50, satir * 100 + 50);
            }
        }
    }
    // Sayı seçme alanı (y=500'den başlar)
    for (let i = 0; i < 4; i++) {
        rect(i * 100, 500, 100, 100);
        text(i + 1, i * 100 + 50, 550);
    }
}

function draw() {
    background(255);
    printBoard(board);
}
```

Oyunun kuralları:

- Oyuncu önce alttaki sayı seçme alanından (1-4) bir rakam seçer. Seçili rakam sayı seçme alanında farklı renkte ya da stilde gözükür.
- Sonra tahtadaki boş bir hücreye tıklayarak seçili rakamı o hücreye yerleştirir.
- Tahta tamamen dolduğunda: tüm satır ve sütunlar geçerliyse **"Kazandınız!"**, değilse **"Kaybettiniz!"** mesajı gösterilir.

İstenen: Önceki sorularda yazdığınız `getSatirSutun()`, `isBoardFull()` ve `isValidBoard()` fonksiyonlarını kullanarak (çağırarak) `mousePressed()` fonksiyonunun içini yazınız ve `printBoard()` fonksiyonunun sonuna yeni satirlar ekleyiniz. *Ayrıca gerek duyarsanınız global değişken ve yeni fonksiyon tanımlayabilirsiniz.*

Cevap:

Soru 7 (SEÇMELİ - B)

10 puan

		3	
		1	
			1
3		**2**	

Soru 1'de verilen başlangıç tahtası yanda gösterilmiştir. Her satır ve her sütunda 1, 2, 3, 4 rakamlarının her birinin tam olarak bir kez bulunması gerektiğini hatırlayarak bu Sudoku bulmacasını çözünüz.

İstenen: Soldaki tahtadaki boş hücreleri doldurunuz.

Soru 8 (SEÇMELİ - B)

15 puan

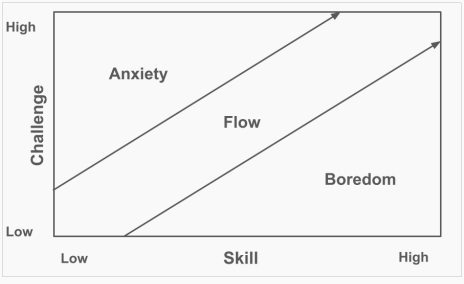
Bu sınavda üzerinde çalıştığımız 4×4 Mini Sudoku, klasik 9×9 Sudoku oyununun basitleştirilmiş bir varyantıdır. Benzer şekilde, Sudoku'nun farklı kurallar veya mekanikler içeren pek çok varyantı bulunmaktadır.

İstenen: Kendi yaratıcılığınızı kullanarak **3 farklı Sudoku varyantı** tasarlayınız. Her varyant için oyunun kurallarını sözel olarak açıklayınız ve Gerekirse şekil/çizim ile destekleyiniz.

Cevap:

Soru 9 (SEÇMELİ - B)

15 puan



Akış Teorisi Diyagramı

Bölümümüzde 15-16 Aralık 2025 tarihlerinde düzenlenen **"Bilişim Günlükleri"** etkinliklerinde, konuşmacı Mustafa Savaş'ın *"Oyun Düşüncesi: Deneyim ve Duygunun Tasarımı"* ve *"Hisset, Tasarla, Oynat"* başlıklı sunumlarında ve izlemenenizin önerildiği eğitsel videoların bazılarında **Akış Teorisi (Flow Theory)** konusuna değinilmişti.

İstenen: Akış Teorisi'nin oyun tasarımındaki önemini birkaç cümle ile açıklayınız.

Cevap: