

**ОТГОВОРИ, УПЪТВАНИЯ И ПРИМЕРНИ ИЗГЛЕДИ
НА РЕШЕНИЯТА НА ПРАКТИЧЕСКИТЕ
ЗАДАЧИ ОТ**

ТЕМА 12

Въпрос	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Отговор	Б	В	А	Б	В	А, В и Г	Г	Б	Г	А	Г	А	В	Б	Б	В	В

18.

SELECT title

FROM movies

WHERE genre = 'Комедия' AND release_year > 2010

ORDER BY release_year;

19. Разликата между въведена дата от потребител и днешната дата, както и съобщение дали разликата е повече, или по-малко от 7 дни.

20.

C#

```
using System;
class Program
{
    static void Main()
    {
        int[,] numbers = { {7, 2, 3 }, { 1, 9, 6 } };
        for (int i = 0; i < numbers.GetLength(1); i++)
        {
            for (int j = 0; j < numbers.GetLength(0); j++)
            {
                Console.Write(numbers[j, i]);
            }
            Console.WriteLine();
        }
    }
}
```

Java

```
public class Main {
    public static void main(String[] args) {
        int[][] numbers = { {7, 2, 3}, {1, 9, 6} };
        for (int i = 0; i < numbers[0].length; i++) {
            for (int j = 0; j < numbers.length; j++) {
                System.out.print(numbers[j][i]);
            }
            System.out.println();
        }
    }
}
```

21.

C#
<pre>using System; class Program { static void Main() { double sum = 0, avg = 0; double[] numbers = { 10, 20, 30, 40 }; for (int i = 0; i < numbers.Length; i++) { sum += numbers[i]; } avg = sum / numbers.Length; Console.WriteLine("Сумата на числата в масива: " + sum); Console.WriteLine("Средната стойност: " + avg); } }</pre>
Java
<pre>public class Main { public static void main(String[] args) { double sum = 0, avg = 0; double[] numbers = { 10, 20, 30, 40 }; for (int i = 0; i < numbers.length; i++) { sum += numbers[i]; } avg = sum / numbers.length; System.out.println("Сумата на числата в масива: " + sum); System.out.println("Средната стойност: " + avg); } }</pre>

22.

- (1) Колко е броят на операциите на изваждане и добавяне? Отговор: 6
(2) Колко са елементите в стека след изпълнението на операциите? Отговор: 0
(3) Кои са елементите на опашката след изпълнението на операциите? Отговор: 5 3 7

23. Б

24. А

25.

C#
<pre>using System; class Program { static void Main() { Console.Write("Въведете първото число: "); int num1 = int.Parse(Console.ReadLine()); } }</pre>

```

        Console.WriteLine("Въведете второто число: ");
        int num2 = int.Parse(Console.ReadLine());

        Console.WriteLine("Въведете третото число: ");
        int num3 = int.Parse(Console.ReadLine());

        int difference = num2 - num1;

        if (num3 - num2 == difference)
        {
            Console.WriteLine($"Числата образуват аритметична
прогресия с разлика {difference} и тя е {(difference > 0 ?
"положителна" : "отрицателна")}.");
        }
        else
        {
            Console.WriteLine("Числата не образуват аритметична
прогресия.");
        }
    }
}

```

Java

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Въведете първото число: ");
        int num1 = scanner.nextInt();

        System.out.print("Въведете второто число: ");
        int num2 = scanner.nextInt();

        System.out.print("Въведете третото число: ");
        int num3 = scanner.nextInt();

        int difference = num2 - num1;

        if (num3 - num2 == difference) {
            System.out.println("Числата образуват аритметична
прогресия с разлика " + difference + " и тя е" + (difference > 0 ?
"положителна" : "отрицателна") + ".");
        } else {
            System.out.println("Числата не образуват аритметична
прогресия.");
        }
    }
}

```

26.

```
CREATE TABLE Professors (  
    ProfessorId INT PRIMARY KEY,  
    FirstName VARCHAR(255),  
    LastName VARCHAR(255),  
    BirthDate DATE,  
    Department VARCHAR(50),  
    AcademicRank VARCHAR(50),  
    Salary DECIMAL(10,2)  
);
```

```
INSERT INTO Professors VALUES  
(1, 'Иван', 'Добрев', '1980-07-27', 'Компютърни науки', 'Асистент', 3000),  
(2, 'Даниел', 'Георгиев', '1976-05-23', 'Химия', 'Доцент', 7000),  
(3, 'Георги', 'Живков', '1982-09-25', 'Компютърни науки', 'Професор', 8000);  
(4, 'Цветелин', 'Александров', '1971-09-21', 'Компютърни науки', 'Професор',  
8000);
```

```
CREATE TABLE Courses (  
    CourseId INT PRIMARY KEY,  
    CourseName VARCHAR(255),  
    InstructorId INT,  
    StartDate DATE,  
    EndDate DATE,  
    Credits INT  
);
```

```
INSERT INTO Courses VALUES  
(101, 'Въведение в бази данни', 1, '2024-03-01', '2024-05-01', 4),  
(102, 'Химия', 2, '2024-02-15', '2024-04-15', 3),  
(103, 'Структури от данни', 3, '2024-03-10', '2024-05-10', 5);  
(104, 'Електронни таблици', 4, '2024-05-09', '2024-06-09', 6);
```

- Извлечете информация за всички професори, които преподават в департамент „Компютърни науки“.

```
SELECT * FROM Professors WHERE Department = 'Компютърни науки';
```

- Брой на курсовете, които имат кредити над 3.

```
SELECT COUNT(*) FROM Courses WHERE Credits > 3;
```

- Извлечете информация за курсовете с кредити над 3 и стартиращи след дата 2024-05-05.

```
SELECT *
FROM Courses
WHERE Credits > 3 AND StartDate > '2024-05-05';
```

- Списък с имената на професорите и техните академични рангове, сортирани по азбучен ред на имената на професорите, за тези от тях, чиито рангове са „Доцент“ или „Професор“.

```
SELECT FirstName, LastName, AcademicRank
FROM Professors
WHERE AcademicRank IN ('Доцент', 'Професор')
ORDER BY FirstName, LastName;
```

27.

C#

```
using System.Collections.Generic;
using System.Linq;
class Product
{
    public string Name { get; set; }
    public string Category { get; set; }
    public double Price { get; set; }
    public int StockQuantity { get; set; }
}
class OnlineStore
{
    private List<Product> products;
    public OnlineStore()
    {
        products = new List<Product>();
    }
    public void AddProduct(Product product)
    {
        products.Add(product);
    }
    public void DisplayProducts(Func<Product, IComparable>
orderBy)
    {
        var sortedProducts = products.OrderBy(orderBy);
        foreach (var product in sortedProducts)
        {
            Console.WriteLine($"Name: {product.Name}, Category:
{product.Category}, Price: {product.Price}, Stock: {product.
StockQuantity}");
        }
    }
    public Product FindProduct(Func<Product, bool> predicate)
    {
        return products.FirstOrDefault(predicate);
    }
}
using System;
public void UpdateProductInfo(string productName,
Action<Product> updateAction)
```

```

    {
        var productToUpdate = FindProduct(p => p.Name ==
productName);
        if (productToUpdate != null)
        {
            updateAction(productToUpdate);
        }
        else
        {
            Console.WriteLine($"Product with name {productName}
not found.");
        }
    }
    public void PurchaseProduct(string productName, int quantity)
    {
        var productToPurchase = FindProduct(p => p.Name ==
productName);
        if (productToPurchase != null)
        {
            if (productToPurchase.StockQuantity >= quantity)
            {
                productToPurchase.StockQuantity -= quantity;
                Console.WriteLine($"Purchase successful.
Remaining stock: {productToPurchase.StockQuantity}");
            }
            else
            {
                Console.WriteLine($"Not enough stock available
for {productName}.");
            }
        }
        else
        {
            Console.WriteLine($"Product with name {productName}
not found.");
        }
    }
    public double CalculateTotalValue()
    {
        return products.Sum(p => p.Price * p.StockQuantity);
    }
}
class Program
{
    static void Main()
    {
        OnlineStore onlineStore = new OnlineStore();
        onlineStore.AddProduct(new Product { Name = "Laptop",
Category = "Electronics", Price = 1200.99, StockQuantity = 10 });
        onlineStore.AddProduct(new Product { Name = "T-shirt",
Category = "Clothing", Price = 19.99, StockQuantity = 50 });
        onlineStore.AddProduct(new Product { Name = "Java Book",
Category = "Books", Price = 39.99, StockQuantity = 20 });
        Console.WriteLine("All Products:");
        onlineStore.DisplayProducts(p => p.Price);
        Console.WriteLine("\nSearch Product by Category:");
    }
}

```

```

        var foundProduct = onlineStore.FindProduct(p =>
p.Category == "Electronics");
        if (foundProduct != null)
        {
            Console.WriteLine($"Found: {foundProduct.Name},
Price: {foundProduct.Price}");
        }
        else
        {
            Console.WriteLine("Product not found.");
        }
        Console.WriteLine("\nUpdate Product Information:");
        onlineStore.UpdateProductInfo("Laptop", p => p.Price =
1299.99);
        onlineStore.DisplayProducts(p => p.Price);
        Console.WriteLine("\nPurchase Product:");
        onlineStore.PurchaseProduct("T-shirt", 5);
        onlineStore.DisplayProducts(p => p.Price);
        Console.WriteLine($" \nTotal Value of Products in the
Store: ${onlineStore.CalculateTotalValue()}");
    }
}

```

Java

```

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
class Product {
    String name;
    String category;
    double price;
    int stockQuantity;
    public Product(String name, String category, double price,
int stockQuantity) {
        this.name = name;
        this.category = category;
        this.price = price;
        this.stockQuantity = stockQuantity;
    }
}
class OnlineStore {
    private List<Product> products;
    public OnlineStore() {
        products = new ArrayList<>();
    }
    public void addProduct(Product product) {
        products.add(product);
    }
    public void displayProducts(Comparator<Product> comparator) {
        products.stream()
            .sorted(comparator)
            .forEach(product ->
                System.out.println("Name: " + product.name
+ ", Category: " + product.category +

```

```

        ", Price: " + product.price + ",
        Stock: " + product.stockQuantity));
    }
    public Product findProduct(java.util.function.
    Predicate<Product> predicate) {
        return products.stream()
            .filter(predicate)
            .findFirst()
            .orElse(null);
    }
    public void updateProductInfo(String productName, java.util.
    function.Consumer<Product> updateAction) {
        Product productToUpdate = findProduct(p -> p.name.
        equals(productName));
        if (productToUpdate != null) {
            updateAction.accept(productToUpdate);
        } else {
            System.out.println("Product with name " + productName
            + " not found.");
        }
    }
    public void purchaseProduct(String productName, int quantity)
    {
        Product productToPurchase = findProduct(p -> p.name.
        equals(productName));
        if (productToPurchase != null) {
            if (productToPurchase.stockQuantity >= quantity) {
                productToPurchase.stockQuantity -= quantity;
                System.out.println("Purchase successful.
                Remaining stock: " + productToPurchase.stockQuantity);
            } else {
                System.out.println("Not enough stock available
                for " + productName + ".");
            }
        } else {
            System.out.println("Product with name " + productName
            + " not found.");
        }
    }
    public double calculateTotalValue() {
        return products.stream()
            .mapToDouble(p -> p.price * p.stockQuantity)
            .sum();
    }
    public static void main(String[] args) {
        OnlineStore onlineStore = new OnlineStore();
        onlineStore.addProduct(new Product("Laptop",
        "Electronics", 1200.99, 10));
        onlineStore.addProduct(new Product("T-shirt", "Clothing",
        19.99, 50));
        onlineStore.addProduct(new Product("Java Book", "Books",
        39.99, 20));
        System.out.println("All Products:");
        onlineStore.displayProducts(Comparator.comparingDouble(p
        -> p.price));
    }
}

```



```

        System.out.println("\nSearch Product by Category:");
        Product foundProduct = onlineStore.findProduct(p ->
p.category.equals("Electronics"));
        if (foundProduct != null) {
            System.out.println("Found: " + foundProduct.name + ",
Price: " + foundProduct.price);
        } else {
            System.out.println("Product not found.");
        }
        System.out.println("\nUpdate Product Information:");
        onlineStore.updateProductInfo("Laptop", p -> p.price =
1299.99);
        onlineStore.displayProducts(Comparator.comparingDouble(p
-> p.price));
        System.out.println("\nPurchase Product:");
        onlineStore.purchaseProduct("T-shirt", 5);
        onlineStore.displayProducts(Comparator.comparingDouble(p
-> p.price));
        System.out.println("\nTotal Value of Products in the
Store: $" + onlineStore.calculateTotalValue());
    }
}

```

28.

C#

```

using System;
class Program
{
    static void Main()
    {
        Console.Write("Enter password: ");
        string password = Console.ReadLine();
        if (IsPasswordValid(password))
        {
            Console.WriteLine("The password is valid.");
        }
        else
        {
            Console.WriteLine("The password is not valid.");
        }
    }
    static bool IsPasswordValid(string password)
    {
        // Проверка за дължина
        if (password.Length < 8)
        {
            return false;
        }
        // Проверка за цифра
        if (!ContainsDigit(password))
        {
            return false;
        }
    }
}

```

```

    }
    // Проверка за малка буква
    if (!ContainsLowercaseLetter(password))
    {
        return false;
    }
    // Проверка за главна буква
    if (!ContainsUppercaseLetter(password))
    {
        return false;
    }
    // Проверка за специални символи
    if (!ContainsSpecialCharacter(password))
    {
        return false;
    }
    return true;
}
static bool ContainsDigit(string password)
{
    return password.ContainsAny("0123456789");
}
static bool ContainsLowercaseLetter(string password)
{
    return password.ContainsAny("abcdefghijklmnopqrstuvwxyz");
}
static bool ContainsUppercaseLetter(string password)
{
    return password.ContainsAny("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
}
static bool ContainsSpecialCharacter(string password)
{
    return password.ContainsAny("!@#$%^&*()-_+=.");
}
}
public static class StringExtensions
{
    public static bool ContainsAny(this string str, string characters)
    {
        foreach (char ch in characters)
        {
            if (str.Contains(ch))
            {
                return true;
            }
        }
        return false;
    }
}
}

```

Java

```
import java.util.Scanner;
public class PasswordValidator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter password: ");
        String password = scanner.nextLine();
        if (isPasswordValid(password)) {
            System.out.println("The password is valid.");
        } else {
            System.out.println("The password is not valid.");
        }
    }
    static boolean isPasswordValid(String password) {
        // Проверка за дължина
        if (password.length() < 8) {
            return false;
        }
        // Проверка за цифра
        if (!containsDigit(password)) {
            return false;
        }
        // Проверка за малка буква
        if (!containsLowercaseLetter(password)) {
            return false;
        }
        // Проверка за главна буква
        if (!containsUppercaseLetter(password)) {
            return false;
        }
        // Проверка за специални символи
        if (!containsSpecialCharacter(password)) {
            return false;
        }
        return true;
    }
    static boolean containsDigit(String password) {
        return password.matches(".*[0-9].*");
    }
    static boolean containsLowercaseLetter(String password) {
        return password.matches(".*[a-z].*");
    }
    static boolean containsUppercaseLetter(String password) {
        return password.matches(".*[A-Z].*");
    }
    static boolean containsSpecialCharacter(String password) {
        return password.matches(".*[!@#$%^&*()-_+=.].*");
    }
}
```