

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

ТЕМА 7

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

17.

C#
<pre>int n; Console.Write("Въведете число: "); n = Int32.Parse(Console.ReadLine()); if (n % 2 == 0) { Console.WriteLine("{0} е четно", n); } else { Console.WriteLine("{0} е нечетно", n); }</pre>
Java
<pre>int n; Scanner scanner = new Scanner(System.in); System.out.print("Въведете число: "); n = scanner.nextInt(); if (n % 2 == 0) { System.out.println(n + " е четно"); } else { System.out.println(n + " е нечетно"); }</pre>

18.

C#
<pre>string sentence; Console.Write("Въведете изречение: "); sentence = Console.ReadLine(); string[] words = sentence.Split(' '); Console.WriteLine("Броят на думите в изречението = " + words.Length);</pre>

Java

```
Scanner scanner = new Scanner(System.in);
System.out.print("Input sentence: ");
String sentence = scanner.nextLine();
scanner.close();
String[] words = sentence.split(" ");
System.out.println("Count = " + words.length);
```

19.

Отговор:

```
CREATE TABLE Courses (
    CourseID INT PRIMARY KEY,
    CourseTitle VARCHAR(255),
    StartDate DATE,
    EndDate DATE,
    InstructorName VARCHAR(100),
    ParticipantCount INT
```

```
);
```

20.

C#

```
using System;
class Program
{
    static void Main()
    {
        Console.Write("Въведете вашата възраст: ");
        int age = int.Parse(Console.ReadLine());
        if (age <= 2)
        {
            Console.WriteLine("Бебе");
        }
        else if (age <= 12)
        {
            Console.WriteLine("Дете");
        }
        else if (age <= 19)
        {
            Console.WriteLine("Тийнейджър");
        }
        else if (age <= 65)
        {
            Console.WriteLine("Възрастен");
        }
        else
        {
            Console.WriteLine("Пенсионер");
        }
    }
}
```

Java

```
import java.util.Scanner;
public class AgeCategory {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Въведете вашата възраст: ");
        int age = scanner.nextInt();
        if (age <= 2) {
            System.out.println("Бебе");
        } else if (age <= 12) {
            System.out.println("Дете");
        } else if (age <= 19) {
            System.out.println("Тийнейджър");
        } else if (age <= 65) {
            System.out.println("Възрастен");
        } else {
            System.out.println("Пенсионер");
        }
    }
}
```

21. А

22.

name	price
четка	3.00
носни кърпички	3.00

23.

status	price
отказана	641
изпратена	126

24.

```
SELECT Teacher, Course, Class
FROM Teachers
INNER JOIN Classes USING (Course)
WHERE Class=12;
```

Или

```
SELECT t.teacher, t.course
FROM Teachers t
JOIN Classes c ON t.course = c.course
WHERE c.class = 12;
```

25.

C#

```

using System;
class Program
{
    static string IsSpecialNumber(int n)
    {
        try
        {
            // Проверка за въведено положително цяло число
            if (n <= 0)
                throw new ArgumentException("Невалидно въведено
число");
            // Проверка дали всички цифри се делят на n
            foreach (char digit in n.ToString())
            {
                if (digit == ',0' || n % (digit - ',0') != 0)
                    return "No";
            }
            return "Yes";
        }
        catch (FormatException)
        {
            return "Something went wrong!";
        }
        catch (ArgumentException)
        {
            return "Something went wrong!";
        }
    }
    static void Main()
    {
        // Въвеждане на число от потребителя
        Console.Write("Въведете цяло положително число: ");
        if (int.TryParse(Console.ReadLine(), out int userInput))
        {
            // Проверка и извеждане на резултата
            string result = IsSpecialNumber(userInput);
            Console.WriteLine(result);
        }
        else
        {
            Console.WriteLine("Something went wrong!");
        }
    }
}

```

Java

```
import java.util.Scanner;
public class Main {
    static String isSpecialNumber(int n) {
        try {
            // Проверка за въведено положително цяло число
            if (n <= 0)
                throw new IllegalArgumentException("Невалидно
въведено число");
            // Проверка дали всички цифри се делят на n
            for (char digit : Integer.toString(n).toCharArray())
            {
                if (digit == ',0' || n % (digit - ',0') != 0)
                    return "No";
            }
            return "Yes";
        } catch (NumberFormatException e) {
            return "Something went wrong!";
        } catch (IllegalArgumentException e) {
            return "Something went wrong!";
        }
    }
    public static void main(String[] args) {
        // Въвеждане на число от потребителя
        Scanner scanner = new Scanner(System.in);
        System.out.print("Въведете цяло положително число: ");
        try {
            int userInput = Integer.parseInt(scanner.nextLine());
            // Проверка и извеждане на резултата
            String result = isSpecialNumber(userInput);
            System.out.println(result);
        } catch (NumberFormatException e) {
            System.out.println("Something went wrong!");
        }
    }
}
```

26.

1. Отговор:

```
CREATE TABLE Exams (
    ID INT PRIMARY KEY,
    Exam_Title VARCHAR(256) NOT NULL,
    Student_Id VARCHAR(30),
    Result DECIMAL(5, 2),
    Exam_Date DATE
);
```

2. Отговори:

Запис 1

```
INSERT INTO Exams (ID, Exam_Title, Student_Id, Result, Exam_Date)
VALUES (1, 'Английски език', '12a-10', 4.50, '2023-12-20');
```

Запис 2

```
INSERT INTO Exams (ID, Exam_Title, Student_Id, Result, Exam_Date)
VALUES (2, 'Английски език', '12b-12', 6.00, '2023-12-27');
```

Запис 3

```
INSERT INTO Exams (ID, Exam_Title, Student_Id, Result, Exam_Date)
VALUES (3, 'Програмиране на Java', '12b-10', 5.25, '2024-01-11');
```

Запис 4

```
INSERT INTO Exams (ID, Exam_Title, Student_Id, Result, Exam_Date)
VALUES (4, 'Алгоритми', '12b-12', 5.00, '2024-01-11');
```

Запис 5

```
INSERT INTO Exams (ID, Exam_Title, Student_Id, Result, Exam_Date)
VALUES (5, 'Структури от данни', '12b-12', 4.75, '2024-01-20');
```

Запис 6

```
INSERT INTO Exams (ID, Exam_Title, Student_Id, Result, Exam_Date)
VALUES (6, 'Програмиране на Java', '12b-12', 5.75, '2024-01-11');
```

3.

```
SELECT Exam_Title, COUNT(DISTINCT Student_Id) AS NumberOfStudents
FROM Exams
GROUP BY Exam_Title;
```

4.

```
SELECT Student_Id, AVG(Result) AS AverageResult
FROM Exams
GROUP BY Student_Id;
```

5.

```
SELECT Exam_Title, Student_Id, Result
FROM Exams
WHERE Exam_Date < '2024-01-01';
```

6.

```
UPDATE Exams
SET Result = 6.00
WHERE Student_Id = '12b-12' AND ID = 6;
```

27.

Таблица „books“

```
CREATE TABLE books (  
    book_id INT PRIMARY KEY,  
    title VARCHAR(255),  
    genre VARCHAR(255),  
    author_id INT,  
    FOREIGN KEY (author_id) REFERENCES authors(author_id)  
);
```

Таблица „authors“

```
CREATE TABLE authors (  
    author_id INT PRIMARY KEY,  
    first_name VARCHAR(255),  
    last_name VARCHAR(255),  
    country VARCHAR(255)  
);
```

Добавяне на данни в **Таблица „authors“**

```
INSERT INTO authors (author_id, first_name, last_name, country)  
VALUES  
    (1, 'George', 'Orwell', 'UK'),  
    (2, 'Oscar', 'Wilde', 'Ireland'),  
    (3, 'Suzanne', 'Collins', 'USA'),  
    (4, 'Emily', 'Brontë', 'UK');
```

Добавяне на данни в **Таблица „books“**

```
INSERT INTO books (book_id, title, genre, author_id)  
VALUES  
    (1, 'The Picture of Dorian Gray', 'Gothic literature', 2),  
    (2, 'The Hunger Games', 'Science fantasy', 3),  
    (3, 'Wuthering Heights', 'Gothic literature', 4),  
    (4, 'Animal Farm', 'Allegory', 1),  
    (5, 'Jane Eyre', 'Gothic literature', 4),  
    (6, '1984', 'Science fantasy', 1);
```

//Напишете заявка, която да извлече данните за всички книги и техните автори.

```
SELECT b.title AS Book_Title, CONCAT(a.first_name, ' ', a.last_name) AS  
Author_Name
```

```
FROM books b
JOIN authors a ON b.author_id = a.author_id;
```

//Напишете заявка, която да извлече данните за всички книги на автора George Orwell.

```
SELECT title
FROM books
WHERE author_id = (
    SELECT author_id
    FROM authors
    WHERE first_name = 'George' AND last_name = 'Orwell'
);
```

//Напишете заявка, която да извлече данните за броя на книгите в жанр Gothic literature.

```
SELECT COUNT(*)
FROM books
WHERE genre = 'Gothic literature';
```

//Напишете заявка, която да извлече данните на всички автори от държава UK.

```
SELECT *
FROM authors
WHERE country = 'UK';
```

//Напишете заявка, която да извлече данните на заглавията на книгите и техните автори, където жанрът е Science fantasy:

```
SELECT b.title AS Book_Title, CONCAT(a.first_name, ' ', a.last_name) AS
Author_Name
FROM books b
JOIN authors a ON b.author_id = a.author_id
WHERE b.genre = 'Science fantasy';
```


C#

```

using System;
using System.Collections.Generic;
class Person
{
    public string Name { get; set; }
    public int Age { get; set; }
    public Person(string name, int age)
    {
        Name = name;
        Age = age;
    }
    public override string ToString()
    {
        return $"Name: {Name}, Age: {Age}";
    }
}
class Student : Person
{
    public int Grade { get; set; }
    public Student(string name, int age, int grade) : base(name,
age)
    {
        Grade = grade;
    }
    public override string ToString()
    {
        return $"Student: {base.ToString()}, Class: {Grade}";
    }
}
class Teacher : Person
{
    public string Subject { get; set; }
    public Teacher(string name, int age, string subject) :
base(name, age)
    {
        Subject = subject;
    }
    public override string ToString()
    {
        return $"Teacher: {base.ToString()}, Subject: {Subject}";
    }
}
class School
{
    public string Name { get; set; }
    public List<Student> Students { get; set; }
    public List<Teacher> Teachers { get; set; }
    public School(string name)
    {

```

```

        Name = name;
        Students = new List<Student>();
        Teachers = new List<Teacher>();
    }
    public void AddStudent(string name, int age, int grade)
    {
        Students.Add(new Student(name, age, grade));
    }
    public void AddTeacher(string name, int age, string subject)
    {
        Teachers.Add(new Teacher(name, age, subject));
    }
    public void DisplayInfo()
    {
        Console.WriteLine($"School: {Name}");
        Console.WriteLine("Students:");
        foreach (var student in Students)
        {
            Console.WriteLine(student);
        }
        Console.WriteLine("Teachers:");
        foreach (var teacher in Teachers)
        {
            Console.WriteLine(teacher);
        }
    }
}
class Program
{
    static void Main()
    {
        Console.Write("Enter the name of the school: ");
        string schoolName = Console.ReadLine();
        School school = new School(schoolName);
        while (true)
        {
            Console.WriteLine($"{school.Name}");
            Console.WriteLine("s - Added a student");
            Console.WriteLine("t - Added a teacher");
            Console.WriteLine("v - View information about the
school");
            Console.WriteLine("q - Exit");
            string choice = Console.ReadLine();
            switch (choice)
            {
                case "s":
                    Console.Write("Name of the student: ");
                    string studentName = Console.ReadLine();

                    Console.Write("Age of the studenta: ");
                    if (int.TryParse(Console.ReadLine(), out int
studentAge))
                    {
                        Console.Write("Student's class: ");

```

```

        if (int.TryParse(Console.ReadLine(), out
int studentGrade))
        {
            school.AddStudent(studentName,
studentAge, studentGrade);
        }
        else
        {
            Console.WriteLine("False: Invalid
age. Try again.");
        }
    }
    else
    {
        Console.WriteLine("False: Invalid class.
Try again.");
    }
    break;
case "t":
    Console.Write("Teacher's name: ");
    string teacherName = Console.ReadLine();

    Console.Write("Age of the teacher: ");
    if (int.TryParse(Console.ReadLine(), out int
teacherAge))
    {
        Console.Write("Subject of the teacher: ");
        string teacherSubject = Console.ReadLine();
        school.AddTeacher(teacherName,
teacherAge, teacherSubject);
    }
    else
    {
        Console.WriteLine("False: Invalid age.
Try again.");
    }
    break;
case "v":
    school.DisplayInfo();
    break;
case "q":
    Environment.Exit(0);
    break;
default:
    Console.WriteLine("Wrong option. Try again.");
    break;
}
}
}
}
}

```

Java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    @Override
    public String toString() {
        return "Name: " + name + ", Age: " + age + " years";
    }
}
class Student extends Person {
    private int grade;
    public Student(String name, int age, int grade) {
        super(name, age);
        this.grade = grade;
    }
    @Override
    public String toString() {
        return "Student: " + super.toString() + ", Grade: " +
grade;
    }
}
class Teacher extends Person {
    private String subject;
    public Teacher(String name, int age, String subject) {
        super(name, age);
        this.subject = subject;
    }
    @Override
    public String toString() {
        return "Teacher: " + super.toString() + ", Subject: " +
subject;
    }
}
class School {
    private String name;
    private List<Student> students;
    private List<Teacher> teachers;
    public School(String name) {
        this.name = name;
        this.students = new ArrayList<>();
        this.teachers = new ArrayList<>();
    }
}
```

```

public void addStudent(String name, int age, int grade) {
    students.add(new Student(name, age, grade));
}
public void addTeacher(String name, int age, String subject)
{
    teachers.add(new Teacher(name, age, subject));
}
public void displayInfo() {
    System.out.println("School: " + name);
    System.out.println("Students:");
    for (Student student : students) {
        System.out.println(student);
    }
    System.out.println("Teachers:");
    for (Teacher teacher : teachers) {
        System.out.println(teacher);
    }
}
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the school name: ");
    String schoolName = scanner.nextLine();
    School school = new School(schoolName);
    while (true) {
        System.out.println(schoolName + " School");
        System.out.println("s - Add student");
        System.out.println("t - Add teacher");
        System.out.println("v - Display school information");
        System.out.println("q - Exit");
        String choice = scanner.nextLine();
        switch (choice) {
            case "s":
                System.out.print("Student's name: ");
                String studentName = scanner.nextLine();
                System.out.print("Student's age: ");
                int studentAge = Integer.parseInt(scanner.
nextLine());
                System.out.print("Student's grade: ");
                int studentGrade = Integer.parseInt(scanner.
nextLine());
                school.addStudent(studentName, studentAge,
studentGrade);
                break;
            case "t":
                System.out.print("Teacher's name: ");
                String teacherName = scanner.nextLine();
                System.out.print("Teacher's age: ");
                int teacherAge = Integer.parseInt(scanner.
nextLine());
                System.out.print("Teacher's subject: ");
                String teacherSubject = scanner.nextLine();

```

```

        school.addTeacher(teacherName, teacherAge,
teacherSubject);
        break;
    case "v":
        school.displayInfo();
        break;
    case "q":
        System.exit(0);
        break;
    default:
        System.out.println("Wrong option. Try again.");
        break;
    }
}
}
}

```