



РЕПУБЛИКА БЪЛГАРИЯ  
Министър на образованието и науката

**ЗА П О В Е Д**

№ РД 09 – ..... г.

На основание чл. 36, ал. 2 от Закона за професионалното образование и обучение, във връзка с чл. 2, ал. 1 и 2 от Наредба № 1 от 19.02.2020 г. за организацията и провеждането на изпитите за придобиване на професионална квалификация, при спазване изискванията на чл. 66, ал. 1 и 2 от Административнопроцесуалния кодекс

**У Т В Ъ Р Ж Д А В А М**

Национална изпитна програма за провеждане на държавен изпит за придобиване на втора степен на професионална квалификация за специалност код **4810101** „Програмно осигуряване“ от професия код **481010** „Програмист“ от професионално направление код **481** „Компютърни науки“.

**X**

КРАСИМИР ВЪЛЧЕВ  
Министър на образованието и науката

*Приложение*

**МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА**

**НАЦИОНАЛНА ИЗПИТНА ПРОГРАМА**

**ЗА ПРОВЕЖДАНЕ НА  
ДЪРЖАВЕН ИЗПИТ ЗА ПРИДОБИВАНЕ  
НА ВТОРА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ**

	<b>Код по СПОО</b>	<b>Наименование</b>
<b>Професионално направление</b>	<b>481</b>	<i>Компютърни науки</i>
<b>Професия</b>	<b>481010</b>	<i>Програмист</i>
<b>Специалност</b>	<b>4810101</b>	<i>Програмно осигуряване</i>

Утвърдена със Заповед № РД 09-.....

София, 2020 г.

**I. ЦЕЛ НА ИЗПИТНАТА ПРОГРАМА**

Националната изпитна програма е предназначена за провеждане на държавния изпит за придобиване на **втора** степен на професионална квалификация по специалност код 4810101 „Програмно осигуряване“, професия код 481010 „Програмист“ от Списъка на

професиите за професионално образование и обучение по чл. 6 от Закона за професионалното образование и обучение.

Целта на настоящата изпитна програма е да определи единни критерии за оценка на професионалните компетентности на обучаваните, изискващи се за придобиване на втора степен по изучаваната професия „Програмист“, специалност „Програмно осигуряване“.

Националната изпитна програма е разработена във връзка с чл. 36 от Закона за професионалното образование и обучение (ЗПОО) и чл. 2, ал. 1 и 2 от Наредба № 1 от 19.02.2020 г. за организацията и провеждането на изпитите за придобиване на професионална квалификация.

## II. ОБЯСНИТЕЛНИ БЕЛЕЖКИ

Националната изпитна програма включва:

- за частта по теория на професията – осемнадесет изпитни теми с кратко описание на учебното съдържание по всяка тема и указание за разработване на писмен тест по всяка изпитна тема;
- за частта по практика на професията - указание за съдържанието на индивидуалните задания;
- критериите за оценяване на резултатите от обучението;
- система за оценяване;
- препоръчителна литература.
- Приложения:
  - а. Примерен изпитен билет;
  - б. Примерно индивидуално задание;
  - в. Примерно указание за разработване на писмен тест.

Държавният изпит – част по теория на професията, се провежда като писмен изпит по една и съща изпитна тема за учениците и/или за обучаваните за дадено училище или обучаваща институция.

Училището/обучаващата институция въз основа на писмено заявено желание на обучаемите по чл. 3, ал. 11 от Наредба № 1 от 19.02.2020 г. за организацията и провеждането на изпитите за придобиване на професионална квалификация може да организира провеждането на държавния изпит – част по теория на професията като писмен тест.

С изпитната тема или изпитния тест се проверява задължителното за усвояване и контрол учебно съдържание на равнища „Знание“, „Разбиране“ и „Приложение“, като броят и равнището на всяка задача се определят към критериите за оценка за всяка изпитна тема.

При избран от училището/обучаващата институция вариант на провеждане на изпита с писмен тест въз основа на критериите за оценка към всяка изпитна тема се съставят тестовите задачи.

Всяка тестова задача задължително съдържа глагол (при възможност започва с глагол), изразяващ действието, което трябва да извърши обучаваният, и показващ равнището по таксономията на Блум, еталона на верния отговор и ключ за оценяване - пълния отговор за който се получават максимален брой точки съобразно равнището на задачата, определени в таблицата за критериите за оценка на всяка изпитна тема.

Към всеки тест се разработва:

1. Указание за работа, която включва:

- целта на теста - какви знания и умения се оценяват с него;
- представяне и описание на теста - брой задачи, типология (задачи със свободен отговор; задачи за допълване/съотнасяне; задачи с избран отговор) и начин на работа с тях;
- продължителност на работа с теста;
- начин на оценяване на резултатите от теста.

2. Методически указания за комисията по оценяване

Всеки член на комисията по оценяване получава тестовите задачи, еталона на верния отговор и ключ за оценяване.

За оценката на писмена работа по изпитна тема комисията по оценяване на изпита – част по теория на професията, назначена със заповед на директора на училището/ръководителя на обучаващата институция, определя за всеки критерий конкретни показатели, чрез които да се диференцира определеният брой присъдени точки.

За оценката на писмения тест комисията използва еталона на верния отговор и ключ за оценяване.

Чрез държавния изпит – част по практика на професията и специалността, се проверяват и оценяват професионалните умения и компетентности на обучаваните, отговарящи на **втора** степен на професионална квалификация. Изпитът се провежда по индивидуални задания и критерии за оценяване, изготвени от комисията за провеждане и оценяване на изпита - част по практика на професията. Броят на изготвените задания трябва да бъде поне с един повече от броя на явяващите се в деня на изпита.

### **III. ИЗПИТНИ ТЕМИ**

*Изпитна тема № 1: Програмиране*

**План-тезис:** Основни понятия: програмиране, език за програмиране, алгоритъм, среда за разработка (IDE), компилация и интерпретация. Пресмятания, оператори, изрази. Условни конструкции. Логически изрази и оператори за сравнение. Вложени условни оператори. Цикли. Вложени цикли. Подпрограми (функции/методи), параметри, връщана стойност.

**Примерна приложна задача:** По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

**Условие:** Разполагате със следния програмен код:

Program.cs

```
age = int.Parse(Console.ReadLine());
washingMachinePrice = int.Parse(Console.ReadLine());
presentPrice = int.Parse(Console.ReadLine());

int savings = 0;
int moneyFromPresents = 0;
int bonus = 0;
for (int i = 1; i < age; i--) {
    if (i % 2 != 0)
        savings += bonus;
        savings -= 1;
        bonus += 10;
    else {
        moneyFromPresents += presentPrice;
    }
}

var allMoney = savings + moneyFromPresents;
if (allMoney == washingMachinePrice)
    Console.WriteLine("Yes! {0:F2}", allMoney - washingMachinePrice);
else
    Console.WriteLine("No! {0:F2}", washingMachinePrice - allMoney);
```

Открийте и поправете грешките във вече написания програмен код, така че да се реши поставената по-долу задача. Допълнете кода.

Лили вече е на **N години**. За всеки свой **рожден ден** тя получава подарък. За **нечетните** рождени дни (**1, 3, 5...n**) получава **игралки**, а за всеки **четен** (**2, 4, 6...n**) получава **пари**. За **втория рожден ден** получава **10.00** лв, като **сумата се увеличава с 10.00** лв, за всеки следващ **четен рожден ден** (**2 ->**

10, 4 -> 20, 6 -> 30...и т.н.). През годините Лили тайно е спестявала парите. Братът на Лили, в годините, които тя получава пари, взима по 1.00 лев от тях. Лили продала играчките получени през годините, всяка за Р лева и добавила сумата към спестените пари. С парите искала да си купи пералня за Х лева. Напишете програма, която да пресмята, колко пари е събрала и дали ѝ стигат да си купи пералня.

#### Вход

От конзолата се прочитат 3 числа, на отделни редове:

- Възрастта на Лили – цяло число в интервала [1...77]
- Цената на пералнята – число в интервала [1.00...10 000.00]
- Единична цена на играчка – цяло число в интервала [0...40]

#### Изход

Да се отпечата на конзолата един ред:

- Ако парите на Лили са достатъчни:
  - “Yes! {N}” – където N е остатъка пари след покупката
- Ако парите не са достатъчни:
  - “No! {M}“ – където M е сумата, която не достига
- Числата N и M трябва да са форматирани до вторият знак след десетичната запетая.

Примерен вход и изход

#### Примери:

Вход	Изход	Коментари
10 170.00 6	Yes! 5.00	Първи рожден ден получава играчка; 2-ри -> 10лв; 3-ти -> играчка; 4-ти -> 10 + 10 = 20лв; 5-ти -> играчка; 6-ти -> 20 + 10 = 30лв; 7-ми -> играчка; 8-ми -> 30 + 10 = 40лв; 9-ти -> играчка; 10-ти -> 40 + 10 = 50лв.

		<p>Спестила е <math>\rightarrow 10 + 20 + 30 + 40 + 50 = 150\text{лв.}</math>          Продала е 5 играчки по 6 лв = 30лв.          Брат ѝ взел 5 пъти по 1 лев = 5лв.      Остават <math>\rightarrow 150 + 30 - 5 = 175\text{лв.}</math>  <math>175 \geq 170</math> (цената на пералнята) успяла е да я купи и са и останали <math>175 - 170 = 5\text{ лв.}</math></p>
<p>21 1570.98 3</p>	No! 997.98	<p>Спестила е 550лв.          Продала е 11 играчки по 3 лв = 33лв. Брат ѝ взимал 10 години по 1 лев = 10лв. Останали <math>550 + 33 - 10 = 573\text{лв}</math>  <math>573 &lt; 1570.98</math> – не е успяла да купи пералня.          Не ѝ достигат <math>1570.98 - 573 = 997.98\text{лв}</math></p>

<b>Критерии за оценяване на изпитна тема № 1</b>	<b>Максимален брой точки</b>
1. Дефинира понятията: програмиране, език за програмиране, алгоритъм, среда за разработка (IDE), компилация и интерпретация.	12
2. Работи с променливи и данни. Създава числови изрази и извършва пресмятания.	12
3. Дефинира и прилага условни конструкции. Обяснява операторите за сравнение, пресмята логически изрази. Дефинира и прилага вложени условни оператори.	18
4. Дефинира и прилага операторите за цикли: for, while, do-while. Дефинира и прилага вложени цикли Обяснява същността и предимствата на подпрограмите (функции/методи). Дефинира и извиква методи. Работа с параметри и върнати стойности.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 2: Програмиране*

**План-тезис:** Команди за работа със сорс-контрол системи. Видове типове данни, бройни системи и понятие за обект. Работа с масиви и списъци. Дебъгване и работа с дебъгер. Символни низове и работа с текст. Многомерни масиви. Речници и хеш-таблицы.

**Условие:**

Разполагате със следния програмен код:

```
Program.cs
var book = new Dictionary <string, string> ();
while (true) {
    var line = Console.ReadLine().Split(' ');
    switch (line[0]) {
        case "A": {
            book[line[1]] = line[2];
            break;
        }
        case "S": {
            if (book.ContainsKey(line[1])) Console.WriteLine("{0} ->
{1}", line[1], book[line[1]]);
            else Console.WriteLine("Contact {0} does not exist.",
line[1]);
            break;
        }
        case "END":
            return;
    }
}
```

Открийте и поправете грешките във вече написания програмен код, така че да решава следната задача:

Напишете програма, която получава информация от конзолата за хора и техните телефонни номера. Всеки запис трябва да има само едно име и телефон (и двете се пазят в низ). На всеки ред ще получите някоя от следните команди:

- A {име} {телефон} = добавя записа към телефонния указател. В случай че се добавя име, което вече съществува в указателя трябва да смените съществуващия номер с новия.
- S {име} = търси се човек с такива име и се извежда резултат във формат "{име} -> {номер}". В случай на не съществуващ контакт, изведете "Contact {име} does not exist."
- END = спира получаването на команди

**Примери:**

Вход	Изход
------	-------



A Minchev 0899148872 S Peter S Minchev END	Contact Peter does not exist. Minchev -> 0899148872
A Peter 112 A Peter 911 S Peter END	Peter 911

<i>Критерии за оценяване на изпитна тема № 2</i>	<i>Максимален брой точки</i>
1. Дефинира понятието сорс-контрол система. Познава начините за работа със сорс-контрол система.	10
2. Описва типове данни.	8
3. Познава бройни системи.	8
4. Умее да работи с масиви и списъци. Различава масиви и списъци.	8
5. Познава начините за работа с текстови низове и текст.	8
6. Дефинира многомерни масиви.	8
7. Знае речници и хеш таблици.	10
8. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

### *Изпитна тема № 3: Обектно-ориентирано програмиране*

**План-тезис:** Дефиниране на класове: клас, конструктор, полета, свойства, създаване на обекти от клас. Дефиниране на функции/методи в класовете, ключова дума **this**. Енкапсулация на данни в класовете, методи за достъп и промяна на полета (getters/setters). Статични полета и методи в класовете.

#### **Примерна приложна задача:**

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

#### **Условие:**

Напишете клас, който съдържа метод, който връща корен квадратен при подадена заявка. Възможно е да получите голям брой заявки, така че трябва да отговаряте бързо на всяка една от тези заявки. Реализирайте *Main()* метод, който да приема едно число – брой на последващите редове. От всеки следващ ред се задава едно цяло число в интервала [1; 1000].

Реализирайте класа *SquareRoot* по следната диаграма:

SquareRoot
- <u>max: int = 1000</u> - <u>values: double[]</u>
- <u>SquareRoot()</u> + <u>Sqrt(value: int) : double</u>

Пример:

Вход	Изход
5	5
25	2.82842712474619
8	1.73205080756888
3	10
100	2
4	

Фрагмент:

SquareRoot.cs
<pre>static class SquareRoot {     private static int max = 1000;     private static double[] values;      static SquareRoot()     {         values = new double[max + 1];         for (int i = 1; i &lt;= max; i++) values[i] = Math.Sqrt(i);     }      public static double Sqrt(int value)     {         return values[value];     } }</pre>

Класът *SquareRoot* трябва да работи със следния програмен фрагмент:

Program.cs
<pre>int n = int.Parse(Console.ReadLine()); while (n &gt; 0) {</pre>

```

int number = int.Parse(Console.ReadLine());
Console.WriteLine(Math.Sqrt(number));
n--;
}

```

<b>Критерии за оценяване на изпитна тема № 3</b>	<b>Максимален брой точки</b>
1. Дефинира: клас, конструктор, полета, свойства, създаване на обекти от клас.	12
2. Дефинира функции/методи в клас.	12
3. Познава ключовата дума <b>this</b> . Енкапсулира данни в класовете. Познава методите за достъп и промяна на полета.	18
4. Работи със статични членове в клас.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

#### *Изпитна тема № 4: **Обектно-ориентирано програмиране***

**План-тезис:** *Шаблонни класове и методи. Наследяване, абстракция и интерфейси. Полиморфизъм.*

#### **Примерна приложна задача:**

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

#### **Условие:**

Създайте абстрактен клас ColoredFigure, който притежава:

- Поле color за отбелязване на цвета (като низ)
- Поле size за отбелязване на размер на фигурата
- Конструктор, който приема за параметри цвят и размер
- Метод Show(), който отпечатва цвета и размера на обекта.
- Абстрактен метод GetName(), който връща името на фигурата
- Абстрактен метод GetArea(), който връща лицето на фигурата

Създайте клас Triangle, който наследява ColoredFigure, като този клас има:

- Конструктор, който извиква конструктора на суперкласа
- Дефиниция за абстрактния метод GetName(), като този метод връща низа "Triangle".
- Дефиниция за абстрактния метод GetArea(), като този метод връща лицето на триъгълника, като триъгълникът се приема за равностранен, със страна size. Използвайте формулата:

$$S = \frac{(size)^2 * \sqrt{3}}{4}$$

Създайте клас Square, който наследява ColoredFigure, като този клас има:

- Конструктор, който извиква конструктора на суперкласа
- Дефиниция за абстрактния метод GetName(), като този метод връща низа "Square".
- Дефиниция за абстрактния метод GetArea(), като този метод връща лицето на квадрата със страна size.

Създайте клас Circle, който наследява ColoredFigure, като този клас има:

- Конструктор, който извиква конструктора на суперкласа
- Дефиниция за абстрактния метод GetName(), като този метод връща низа "Circle".
- Дефиниция за абстрактния метод GetArea(), като този метод връща лицето на кръга, с радиус size.

#### Вход:

На първия ред на входа има единствено цяло число N – брой заявки. От следващите N реда се подава заявка в един от следните формати:

- Triangle <цвет> <размер>
- Circle <цвет> <размер>
- Square <цвет> <размер>

#### Изход:

За всяка заявка трябва да създаде обект от съответния клас, след което трябва да изпечатате 4 реда:

<име на фигурата>:

Color: <цвет>

Size: <размер>

Area: <лице>

Отпечатайте лицето с точно два знака след запетаята.

Пример:

Вход	Исход
3 Circle blue 1 Square red 2 Triangle green 3	Circle: Color: blue Size: 1 Area: 3.14 Square: Color: red Size: 2 Area: 4.00 Triangle: Color: green Size: 3 Area: 7.79

Фрагмент:

ColoredFigure.cs

```
public abstract class ColoredFigure {
    protected string color;

    protected double size;

    public ColoredFigure(string color, double size) {
        this.color = color;
        this.size = size;
    }

    public void Show() {
        Console.WriteLine(string.Format("Color: {0}", this.color));
        Console.WriteLine(string.Format("Size: {0}", this.size));
    }

    public abstract string GetName();

    public abstract double GetArea();
}
```

Circle.cs

```
public class Circle: ColoredFigure {
    public Circle(string color, double size): base(color, size) { }

    public override double GetArea() {
        return Math.PI * Math.Pow(base.size, 2.0);
    }

    public override string GetName() {
        return "Circle";
    }
}
```

Triangle.cs

```
public class Triangle: ColoredFigure {
    public Triangle(string color, double size): base(color, size) {}

    public override double GetArea() {
        return (Math.Pow(base.size, 2.0) * Math.Sqrt(3)) / 4.0;
    }

    public override string GetName() {
        return "Triangle";
    }
}
```

Square.cs

```
public class Square: ColoredFigure {
    public Square(string color, double size): base(color, size) {}

    public override double GetArea() {
        return Math.Pow(base.size, 2.0);
    }

    public override string GetName() {
        return "Square";
    }
}
```

Program.cs

```

class Program {
    static void Main(string[] args) {
        int n = int.Parse(Console.ReadLine());
        for (int i = 0; i < n; i++) {
            var cmd = Console.ReadLine().Split();
            switch (cmd[0]) {
                case "Circle": {
                    Circle circle = new Circle(cmd[1],
double.Parse(cmd[2]));
                    Console.WriteLine(string.Format("Name: {0}",
circle.GetName()));
                    circle.Show();
                    Console.WriteLine(string.Format("Area: {0:f2}",
circle.GetArea()));
                    break;
                }
                case "Square": {
                    Square square = new Square(cmd[1],
double.Parse(cmd[2]));
                    Console.WriteLine(string.Format("Name: {0}",
square.GetName()));
                    square.Show();
                    Console.WriteLine(string.Format("Area: {0:f2}",
square.GetArea()));
                    break;
                }
                case "Triangle": {
                    Triangle triangle = new Triangle(cmd[1],
double.Parse(cmd[2]));
                    Console.WriteLine(string.Format("Name: {0}",
triangle.GetName()));
                    triangle.Show();
                    Console.WriteLine(string.Format("Area: {0:f2}",
triangle.GetArea()));
                    break;
                }
            }
        }
    }
}

```

<i>Критерии за оценяване на изпитна тема № 4</i>	<i>Максимален брой точки</i>
1. Познава и прилага концепцията типизиране на класове, чрез шаблонни класове и методи.	12

2. Познава концепцията за наследяване и абстракция.	12
3. Различава презаписване (override) и презареждане (overload) на метод	12
4. Различава клас и интерфейс.	12
5. Разбира концепцията за полиморфизъм.	12
6. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 5: **Обектно-ориентирано програмиране***

**План-тезис:** Итератори. Компаратори. Ламбда изрази и функции. Библиотека за обработка на колекции. Изключения. Работа с потоци и файлове.

**Примерна приложна задача:**

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

**Условие:**

Напишете програма, която моделира 2 превозни средства (**Car** и **Truck**). Трябва да може да симулирате **шофиране** и **презареждане** на превозните средства. **Car** и **truck** имат **количество гориво**, **консумация на гориво в литри за км** и могат да бъдат **управлявани на дадено разстояние** и **презаредени с определено количество гориво**. Но през лятото и двете превозни средства използват климатик и тяхната **консумация** за км е завишена с **0.9** литра за **Car** и с **1.6** литра за **Truck**. Също така **камионът** има малка дупка в резервоара и когато се **зарежда** получава само **95%** от **горивото**. **Колата** няма проблеми със зареждането и получава всичкото гориво. Ако превозното средство не може да измине даденото разстояние, горивото му не се променя.

**Вход:**

- На **първи ред** – информация за колата във формат **{Car {fuel quantity} {liters per km}}**
- На **втори ред** – информация за камиона във формат **{Truck {fuel quantity} {liters per km}}**
- На трети ред – **брой команди N**, които ще бъдат подадени на следващите N реда
- На следващите N реда – команди във формат:
  - **Drive Car {distance}**
  - **Drive Truck {distance}**
  - **Refuel Car {liters}**
  - **Refuel Truck {liters}**

**Изход:**



След всяка **Drive** команда отпечатайте дали колата/камионът може да пропътува разстоянието, като използвате следния формат при успех:

*Car/Truck travelled {distance} km*

Или при неуспех:

*Car/Truck needs refueling*

Накрая отпечатайте **оставащото гориво** за колата и камиона закръглено до **2** знака след запетаята във формат:

*Car: {liters}*

*Truck: {liters}*

Пример:

Вход	Изход
Car 15 0.3 Truck 100 0.9 4 Drive Car 9 Drive Car 30 Refuel Car 50 Drive Truck 10	Car travelled 9 km Car needs refueling Truck travelled 10 km Car: 54.20 Truck: 75.00
Car 30.4 0.4 Truck 99.34 0.9 5 Drive Car 500 Drive Car 13.5 Refuel Truck 10.300 Drive Truck 56.2 Refuel Car 100.2	Car needs refueling Car travelled 13.5 km Truck needs refueling Car: 113.05 Truck: 109.13

Фрагмент:

```
IVehicle.cs
```

```
interface IVehicle
{
    double Fuel { get; }
    double Litersperkm { get;}
    double Distance { get;}
    void Drive(double km);
    void Refuel(double litres);
}
```

```
Car.cs
```

```

class Car : IVehicle
{
    private double fuel, litersperkm, distance;
    public double Fuel { get { return fuel; } }
    public double Litersperkm { get { return litersperkm; } }
    public double Distance { get { return distance; } }

    public void Drive(double km)
    {
        double travel = litersperkm * km;
        if (travel <= fuel)
        {
            fuel -= travel;
            Console.WriteLine($"Car travelled {km} km");
            distance += km;
        }
        else
        {
            Console.WriteLine("Car needs refueling");
        }
    }

    public void Refuel(double litres)
    {
        this.fuel += litres;
    }
    public Car(double fuel, double l)
    {
        this.fuel = fuel;
        this.litersperkm = l+0.9;
    }
}

```

Truck.cs

```

class Truck:IVehicle
{
    private double fuel, litersperkm, distance;
    public double Fuel { get { return fuel; } }
    public double Litersperkm { get { return litersperkm; } }
    public double Distance { get { return distance; } }

    public void Drive(double km)
    {
        double travel = litersperkm * km;
        if (travel <= fuel)
        {

```

```

        fuel -= travel;
        Console.WriteLine($"Truck travelled {km} km");
        distance += km;
    }
    else
    {
        Console.WriteLine("Truck needs refueling");
    }
}

public void Refuel(double litres)
{
    this.fuel += litres;
}
public Truck(double fuel, double l)
{
    this.fuel = fuel*0.95;
    this.litersperkm = l + 1.6;
}
}

```

Program.cs

```

static void Main(string[] args)
{
    var input =
    Console.ReadLine().Split().Skip(1).Select(double.Parse).ToArray();
    Car car = new Car(input[0], input[1]);
    input =
    Console.ReadLine().Split().Skip(1).Select(double.Parse).ToArray();
    Truck truck = new Truck(input[0], input[1]);
    int n = int.Parse(Console.ReadLine());
    for (int i = 0; i < n; i++)
    {
        var cmd = Console.ReadLine().Split().ToArray();
        if (cmd[0] == "Drive")
        {
            if (cmd[1] == "Car")
            {
                car.Drive(double.Parse(cmd[2]));
            }
            else
            {
                truck.Drive(double.Parse(cmd[2]));
            }
        }
    }
}

```

```

else
{
    if (cmd[1] == "Car")
    {
        car.Refuel(double.Parse(cmd[2]));
    }
    else
    {
        truck.Refuel(double.Parse(cmd[2]));
    }
}

}
Console.WriteLine($"Car: {car.Fuel:F2}");
Console.WriteLine($"Truck: {truck.Fuel:f2}");
}

```

<b>Критерии за оценяване на изпитна тема № 5</b>	<b>Максимален брой точки</b>
1. Различава итератори и компаратори. Разбира и дава примери за приложението на итераторите и компараторите.	12
2. Познава ламбда изрази и функции. Обяснява начините за описване на различни ламбда изрази и функции.	12
3. Познава техники за работа с библиотека за поточна ( <i>fluent</i> ) обработка на колекции.	18
4. Описва: референции към методи/функции, изключения, работа с потоци и файлове.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 6: Алгоритми и структури от данни.*

*План-тезис: Въведение в алгоритмите. Линейни структури от данни. Списък, стекове, опашки и имплементации. Алгоритми върху линейни структури: подредици, нарастващи редици, площадка от еднакви елементи.*

**Примерна приложна задача:**

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

**Условие:**

Имплементирайте динамичен свързан списък. Списъкът трябва да поддържа следните операции:

- Добавяне на елемент
- Премахване на елемент по индекс
- Премахване на елемент по референция
- Намиране на индекса на елемент по референция
- Връщане на текущия брой на елементите в списъка
- Индексатор

**Фрагмент:**

**Node.cs**

```
public class Node {
    private object element;
    public object Element {
        get {
            return this.element;
        }
        set {
            this.element = value;
        }
    }
}

private Node next;
public Node Next {
    get {
        return this.next;
    }
    set {
        this.next = value;
    }
}

public Node(object element, Node prevNode) {
    this.element = element;
    prevNode.next = this;
}

public Node(object element) {
    this.element = element;
    next = null;
}
}
```

## DynamicList.cs

```
public class DynamicList {
    private Node head;
    private Node tail;

    private int count;
    public int Count {
        get {
            return this.count;
        }
        private set {
            this.count = value;
        }
    }

    public DynamicList() {
        this.head = null;
        this.tail = null;
        this.count = 0;
    }

    public void Add(object item) {
        if (this.head == null) {
            Node next = new Node(item);
            this.head = next;
            this.tail = next;
        } else {
            Node next = new Node(item, tail);
            this.tail = next;
        }
        this.count++;
    }

    public int IndexOf(object item) {
        Node current = head;
        int index = 0;
        while (current != null) {
            if (current.Element.Equals(item)) return index;
            index++;
            current = current.Next;
        }
        return -1; // Not Found
    }

    public object Remove(int index) {
        Object item = null;
        Node current = head;
        Node previous = null;
```

```

int i = 0;
while (current != null) {
    if (index == i) {
        item = current.Element;
        previous.Next = current.Next;
        break;
    }
    i++;
    previous = current;
    current = current.Next;
}
return item;
}

public int Remove(object item) {
    int index = 0;
    Node current = head;
    while (current != null) {
        if (current.Element.Equals(item)) {
            Remove(index);
            return index;
        }
        index++;
        current = current.Next;
    }
    return -1; // Not Found
}

public object this[int index] {
    get {
        Node current = head;
        int i = 0;
        while (current != null) {
            if (i == index) return current.Element;
            i++;
            current = current.Next;
        }
        return null; // Not Found
    }
    set {
        Node current = head;
        int i = 0;
        while (current != null) {
            if (i == index) {
                current.Element = value;
                break;
            }
            i++;
            current = current.Next;
        }
    }
}

```

```
}  
  }  
    }  
      }
```

<i>Критерии за оценяване на изпитна тема № 6</i>	<i>Максимален брой точки</i>
1. Познава имплементацията на линейни структури от данни от тип списък. Познава и сравнява видовете списъци според начина на имплементация.	12
2. Познава имплементацията на линейни структури от данни от тип стек.	12
3. Познава имплементацията на линейни структури от данни от тип опашка. Различава и сравнява стек и опашка.	18
4. Описва алгоритми върху линейни структури: подредици, нарастващи редици, площадка от еднакви елементи.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>



## Изпитна тема № 7: Алгоритми и структури от данни

**План-тезис:** Сортиране и търсене. Сортиране, устойчивост, бързи и бавни алгоритми. Метод на пряката селекция, метод на мехурчето, сортиране чрез вмъкване, сортиране чрез броене, бързо сортиране, сортиране чрез сливане и имплементации. Линеино търсене, двоично търсене, интерполационно търсене.

### Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

### Условие:

Напишете програма, която сортира таблица от числови стойности по произволна колона.

### Вход:

- Входните данни трябва да се прочетат от конзолата.
- На първия ще ви бъдат подадени три цели числа  $R$ ,  $C$  и  $S$ , разделени с интервал.  $R$  е броят на редовете в таблицата,  $C$  - броят на колоните, а  $S$  - номерът на колоната, по която трябва да бъде сортиран масива.
- На следващите  $R$  реда ще са числата от всеки от редовете в таблицата,  $C$  на брой, разделени с интервали.
- Входните данни винаги ще са валидни и в описания формат

### Изход:

- Изходните данни трябва да бъдат отпечатани на конзолата.
- На  $R$  реда трябва да бъдат изведени числата от таблицата, сортирани по указаната колона.

### Подсказки:

- Ще се наложи да ползвате двумерен масив или масив от масиви.
- Тествайте задачата с различно големи масиви и различни алгоритми за сортиране.
- Обърнете внимание дали при стабилни и нестабилни алгоритми се получава еднакъв резултат.
- Коментирайте в клас има ли начин за минимизиране на размяната на редовете при сортирането.

### Пример 1:

Вход	Изход
3 4 1	1 2 3 4
1 2 3 4	2 3 1 2
3 1 2 4	3 1 2 4
2 3 1 2	

*Коментар към пример 1:* Има 3 реда и 4 колони. Вторият и третият ред трябва да разменят местата си, ако сортираме по колона 1.

Пример 2:

Вход	Изход
4 2 2	3 1
1 2	1 2
3 1	2 3
2 3	4 4
4 4	

*Коментар към пример 2: Има 4 реда и 2 колони. Първият и вторият ред трябва да разменят местата си, ако сортираме по колона 2.*

Фрагмент:

```
Program.cs
if (number == 5)
{
    var input = Console.ReadLine().Split(' ');
    var d2arr = new int[input[0]][];
    var sort = new int[input[1]];
    for (int r = 0; r < input[0]; r++)
    {
        d2arr[r] = new int[input[1]];
        var arr = Console.ReadLine().Split(' ');
        for (int c = 0; c < input[1]; c++)
            d2arr[r][c] = int.Parse(arr[c]);
    }
    Sorting.Sort2D(d2arr, input[2] - 1);
    for (int r = 0; r < input[0]; r++)
        Console.WriteLine(string.Join(" ", d2arr[r]));
}
```

<b>Критерии за оценяване на изпитна тема № 7</b>	<b>Максимален брой точки</b>
1. Различава и сравнява алгоритми за сортиране и търсене. Дефинира понятията устойчивост и сложност на алгоритмите.	12
2. Разграничава различните алгоритми за сортиране.	12
3. Познава имплементацията на алгоритмите за сортиране.	18
4. Различава и сравнява алгоритми за линейно търсене, двоично търсене и интерполационно търсене.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 8: **Базы от данни***

***План-тезис:** Въведение в базите от данни. Типове данни. Основни команди. Моделиране на релационни бази от данни. Връзки между таблици. Ограничения в таблици.*

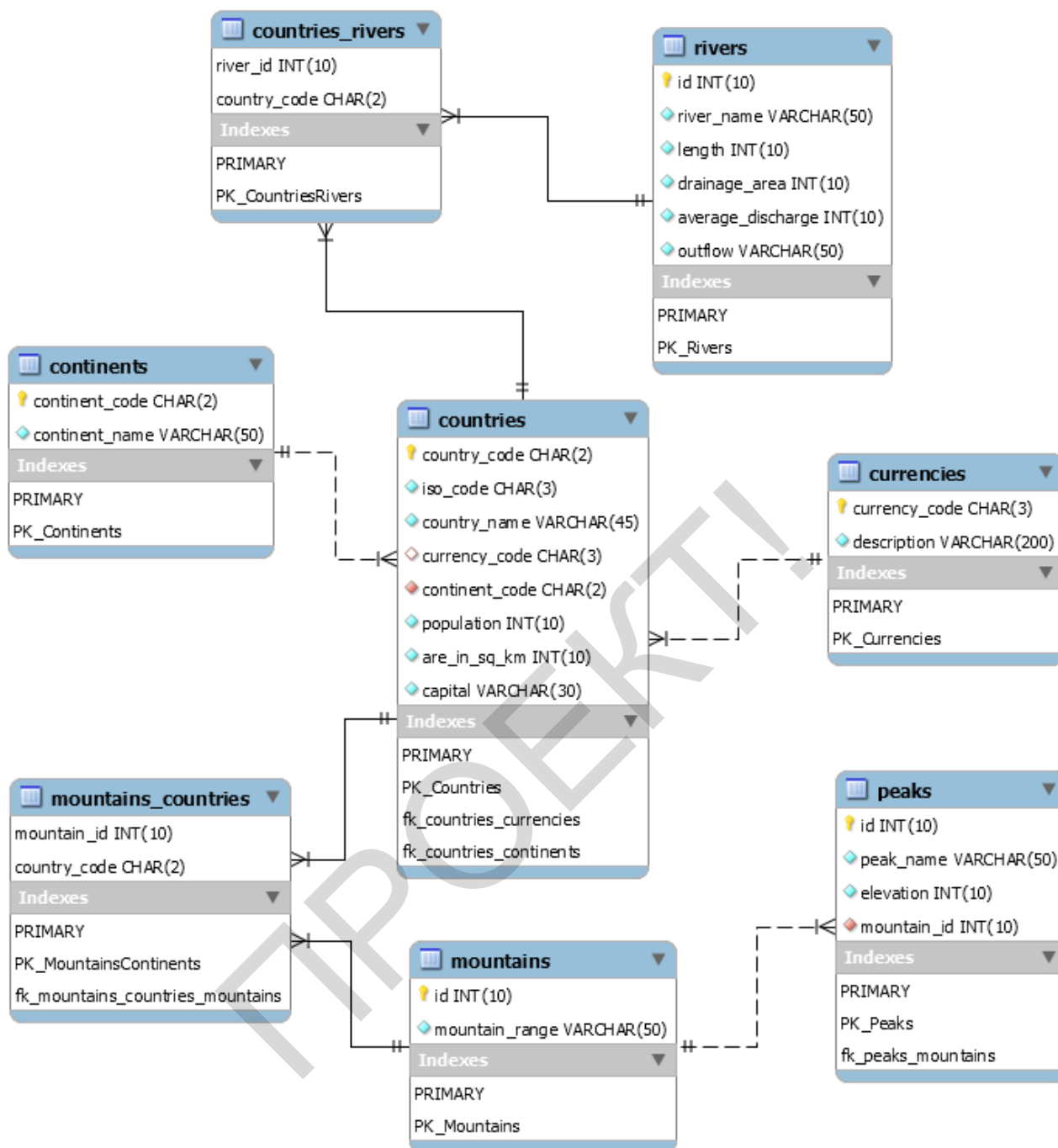
Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Като използвате дадената ER диаграма напишете заявките за създаване на базата данни по-долу:

ПРОЕКТ!



<i>Критерии за оценяване на изпитна тема № 8</i>	<i>Максимален брой точки</i>
1. Описва типове данни и основни команди при работа с бази от данни.	12
2. Познава заявки за дефиниране на таблици (DDL SQL).	12
3. Описва видове връзки между таблици в база данни	18
4. Познава видовете ограничения в база данни	18

5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 9: **Бази от данни***

***План-тезис:** Заявки за извличане и промяна на данни. Вложени заявки. Обединение на заявки.*

*Съединения на заявки за извличане на данни от бази данни. Извличане на части от таблица/и и задаване на псевдоними.*

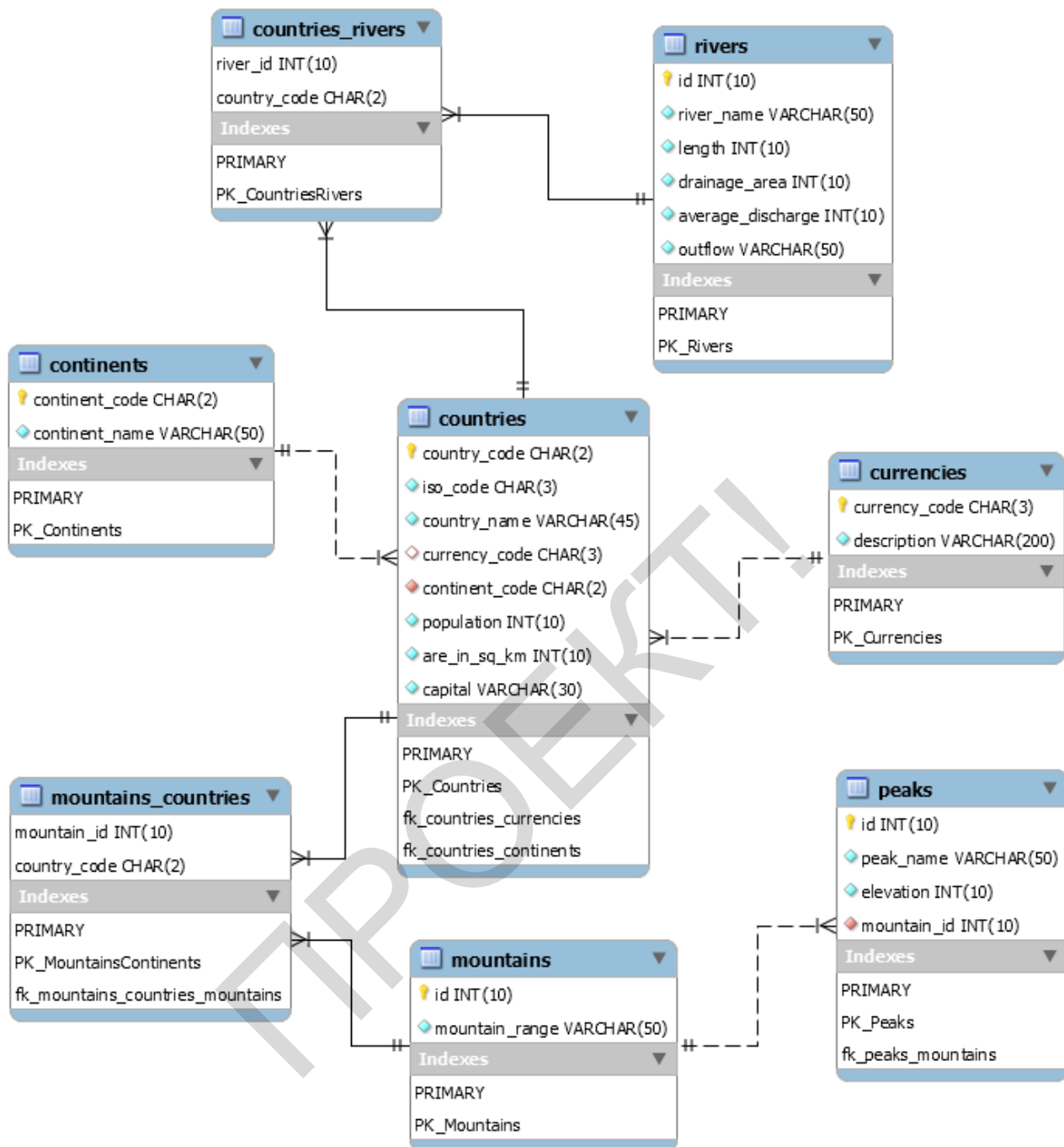
Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Като използвате дадената ER диаграма напишете заявките дадени по-долу:

ПРОЕКТ!



### Заявка 1. Всички планински върхове

Напишете заявка, която всички планински върхове в азбучен ред.

Пример:

peak_name
Aconcagua
Banski Suhodol
Batashki Snezhnik

...

Фрагмент:

query1.sql

```
SELECT peak_name FROM `geography`.`peaks` ORDER BY peak_name;
```

### Заявка 2. Най-големи по население страни

Намерете 30 най-големи по население страни в Европа. Покажете името на страната и населението. Сортирайте резултатите по население в намаляващ ред, след това по страна по азбучен ред

Пример:

country_name	population
Russia	140702000
Germany	81802257
France	64768389
...	...

Фрагмент:

query2.sql

```
SELECT `country_name`,`population` FROM geography.countries  
WHERE `continent_code` = 'EU'  
ORDER BY `population` DESC  
LIMIT 30;
```

### Заявка 3. Най-високи върхове в България

Намерете всички страни заедно с информация за съответната валута. Покажете името на страната, код на страната и информация за валутата ѝ: "Euro" или "Not euro". Сортирайте резултатите по име на страната по азбучен ред.

Пример:

country_name	country_code	currency
Afghanistan	AF	Not Euro

Åland	AX	Euro
Albania	AL	Not Euro
...	...	...

Фрагмент:

```
query3.sql
(
  SELECT country_name, country_code, 'EURO' as 'currency'
  FROM geography.countries
  WHERE currency_code = 'EUR'
) UNION (
  SELECT country_name, country_code, 'NOT EURO' as 'currency'
  FROM geography.countries
  WHERE currency_code <> 'EUR'
)
ORDER BY country_name ASC;
```

<b>Критерии за оценяване на изпитна тема № 9</b>	<b>Максимален брой точки</b>
1. Познава заявките за извличане на данни.	12
2. Познава различните видове съединения и вложени заявки.	12
3. Описва заявки за промяна на съдържанието на таблица в база данни	18
4. Умее да създава заявки за извличане на части от таблици и да използва псевдоними	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 10: Базы от данни*

*План-тезис: Агрегирани функции. Групиране на данни. Скаларни функции, транзакции, съхранени процедури, тригери.*

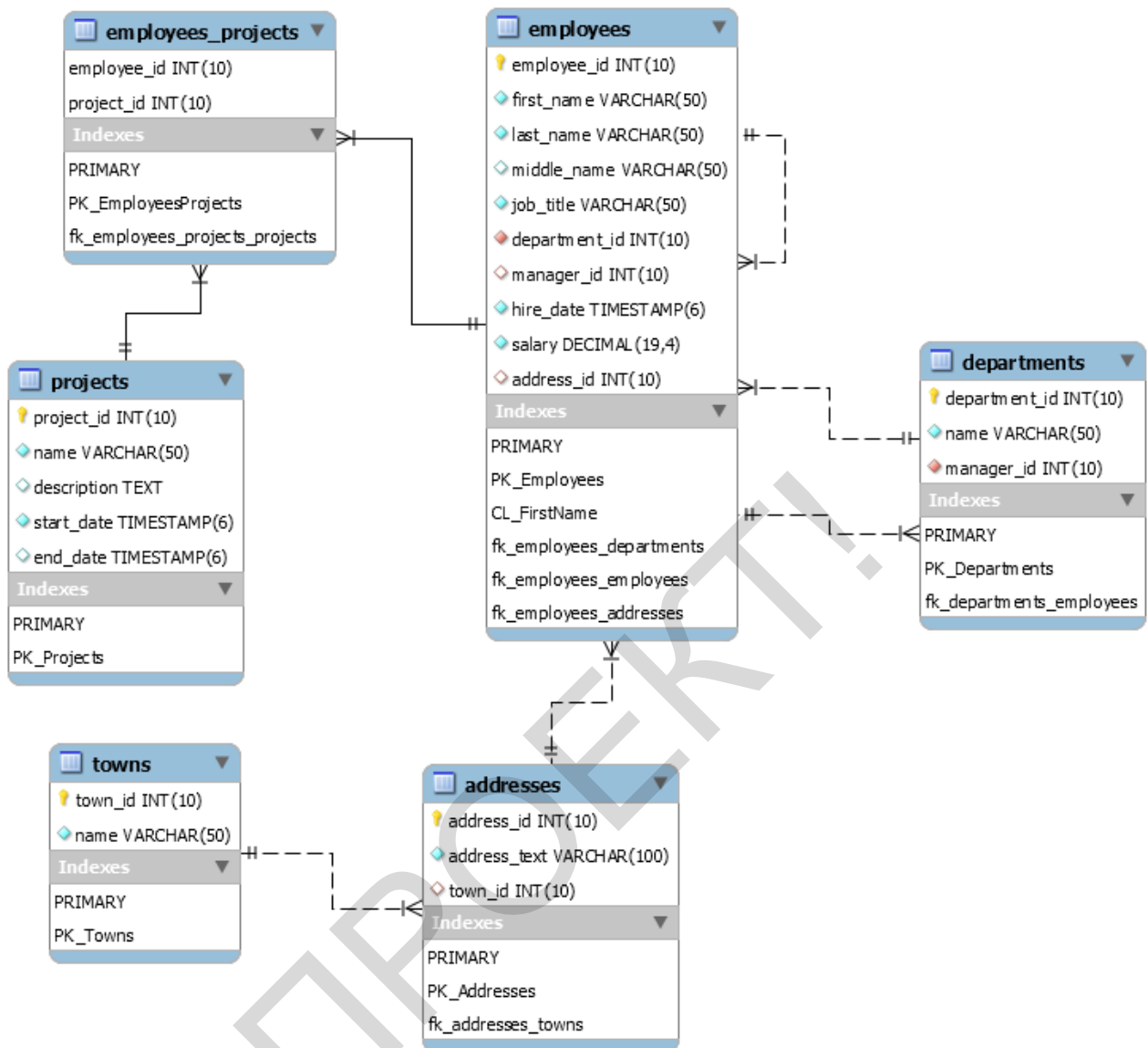
Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:



Като използвате дадената ER диаграма напишете заявките дадени по-долу:



### Заявка 1. Най-висока заплата по длъжности

Напишете заявка, която показва най-високата заплата, давана на служител за всяка длъжност.

Списъкът да е подреден по заплати в низходящ ред и по длъжност, в азбучен.

Пример:

job_title	salary
Chief Executive Officer	125500.0000
Vice President of Production	84100.0000

Vice President of Sales	72100.0000
Vice President of Engineering	63500.0000
...	...

Фрагмент:

```
query1.sql
SELECT job_title, salary
FROM employees AS e
WHERE salary =
(
  SELECT salary
  FROM employees AS emp
  WHERE e.department_id=emp.department_id
  ORDERBY salary DESC LIMIT 1
)
ORDER BY salary DESC, job_title;
```

**Заявка 2. Най-ниско платени служители по отдели**

Напишете заявка, която извежда името и фамилията, заплатата и името на отдела на всички служители, получаващи най-ниска заплата в своя отдел. Резултатът да е сортиран по заплата, име и фамилия, във възходящ ред.

Пример:

first_name	last_name	salary	department
Jimmy	Bischoff	9000.0000	Shipping and Receiving
Kim	Ralls	9000.0000	Shipping and Receiving
Susan	Eaton	9000.0000	Shipping and Receiving
Jo	Berry	9300.0000	Facilities and Maintenance
...	...	...	...

Фрагмент:

```
query2.sql
SELECT first_name, last_name, salary, (SELECT `name` FROM departments AS eee WHERE
e.department_id=eee.department_id LIMIT 1) department_name
FROM employees AS e
```

```

WHERE salary =
(
  SELECT salary
  FROM employees AS emp
  WHERE e.department_id=emp.department_id
  ORDER BY salary LIMIT 1
)
ORDER BY salary, first_name, last_name;

```

### Заявка 3. Мениджъри с по-ниска заплата

Напишете заявка, извеждаща списък на всички мениджъри с поне един подчинен, който има по-висока заплата от тяхната.

Пример:

first_name	last_name
Roberto	Tamburello
Peter	Krebs

Фрагмент:

query3.sql
<pre> SELECT first_name, last_name FROM employees AS e WHERE employee_id = ANY (   SELECT manager_id   FROM employees AS emp   WHERE e.salary &lt; emp.salary ) ORDER BY first_name DESC; </pre>

### Заявка 4. Мениджъри с точно 5 подчинени

Изведете името и фамилията на всички мениджъри с точно 5 подчинени.

Пример:

first_name	last_name
Cristian	Petculescu
Jeff	Hay
Katie	McAskill-White

Lori	Kane
Paula	Barreto de Mattos
Pilar	Ackerman
Shane	Kim
Zheng	Mu

Фрагмент:

```
query4.sql
SELECT e.`first_name`, e.`last_name`
FROM `employees` AS e
WHERE e.`employee_id` IN (SELECT DISTINCT manager_id FROM `employees`)
  AND (SELECT `manager_id` FROM `employees` WHERE `manager_id` = e.`employee_id`
LIMIT 4,1) IS NOT NULL
  AND (SELECT `manager_id` FROM `employees` WHERE `manager_id` = e.`employee_id`
LIMIT 5,1) IS NULL
ORDER BY `first_name` ASC, `last_name` ASC;
```

<b>Критерии за оценяване на изпитна тема № 10</b>	<b>Максимален брой точки</b>
1. Познава типове данни и основни команди при работа с бази от данни.	12
2. Определя моделите на релационни бази от данни. Познава заявки за извличане и промяна на данни.	12
3. Разбира сложни заявки за извличане на данни, съединения на таблици, агрегиращи функции и групиране на данни.	18
4. Описва скаларни функции, транзакции, съхранени процедури и тригери.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 11: **Разработка на софтуер***

*План-тезис: Трислоен модел и MVC. Концепция за тестване и писане на компонентни тестове. Концепция за дебъгване, откриване и отстраняване на грешки. Концепция за рефакториране и правене на „инкрементални промени“.*

Примерна приложна задача:

Условие:

Разполагате с класове `Dummy` и `Axe`. Напишете компонентни тестове, които да проверят работата на класовете и да помогнат в откриването и отстраняването на евентуални проблеми.

За класът `Axe` създайте следните тестове:

- Тествайте дали оръжието губи здравина след всяка атака
- Тествайте атака със счупено оръжие

За класът `Dummy` създайте следните тестове:

- Чучелото губи здраве, ако е атакувано
- Мъртво чучело хвърля изключение, ако е атакувано
- Мъртвото чучело може да даде XP
- Живото чучело не може да даде XP

Фрагмент:

Axe.cs

```
public class Axe
{
    private int attackPoints;
    private int durabilityPoints;

    public Axe(int attack, int durability)
    {
        this.attackPoints = attack;
        this.durabilityPoints = durability;
    }

    public int AttackPoints
    {
        get { return this.attackPoints; }
    }

    public int DurabilityPoints
    {
        get { return this.durabilityPoints; }
    }

    public void Attack(Dummy target)
    {
        if (this.durabilityPoints <= 0)
        {
            throw new InvalidOperationException("Axe is broken.");
        }
    }
}
```

```
    }  
    target.TakeAttack(this.attackPoints);  
    this.durabilityPoints -= 1;  
} }  
}
```

Dummy.cs

```
public class Dummy  
{  
    private int health;  
    private int experience;  
  
    public Dummy(int health, int experience)  
    {  
        this.health = health;  
        this.experience = experience;  
    }  
  
    public int Health  
    {  
        get { return this.health; }  
    }  
  
    public void TakeAttack(int attackPoints)  
    {  
        if (this.IsDead())  
        {  
            throw new InvalidOperationException("Dummy is dead.");  
        }  
  
        this.health -= attackPoints;  
    }  
  
    public int GiveExperience()  
    {  
        if (!this.IsDead())  
        {  
            throw new InvalidOperationException("Target is not dead.");  
        }  
  
        return this.experience;  
    }  
  
    public bool IsDead()  
    {  
        return this.health <= 0;  
    }  
}
```

	<i>брой точки</i>
1. Описва трислоен модел и MVC.	12
2. Разбира концепция за тестване и писане на компонентни тестове.	12
3. Познава концепция за дебъгване, откриване и отстраняване на грешки.	18
4. Знае концепции за рефакториране и правене на „инкрементални промени“.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

### *Изпитна тема № 12: **Разработка на софтуер***

*План-тезис: Инструменти за разработка. Техники за продуктивно използване на интегрирана среда за разработка. Използване на външни библиотеки. Управление на пакети.*

#### Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

#### Условие:

Да се напише програма, която трябва да обработва информация за филми в JSON вид.

Създайте клас `Movie` със свойства за:

- `Id (int)` – номер на филма
- `Name (string)` – име на филма
- `Rating (decimal)` – IMDb рейтинг на филма
- `Year (DateTime)` – година на премиерата на филма

#### Подзадачи:

- Преобразувайте данните от клас `Movie` към JSON
- Преобразувайте JSON към `Movie`

Изберете подходяща външна библиотека за работа с JSON за реализиране на подзадачите.

#### Фрагмент:

```
Movie.cs
```

```

public class Movie
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Rating { get; set; }
    public DateTime Year { get; set; }
}

```

Program.cs

```

// part 1
List<Movie> list1 = new List<Movie>();
list1.Add(new Movie()
{
    Id = 1,
    Name = "Iron Man",
    Rating = 7.9m,
    Year = new DateTime(2008)
});
list1.Add(new Movie()
{
    Id = 2,
    Name = "Monsters Inc.",
    Rating = 8.0m,
    Year = new DateTime(2001)
}); ;
var json1 = JsonConvert.SerializeObject(list1);
Console.WriteLine(json1);

// part 2
var json2 = @"[{Id:1,Name:Iron Man,Rating:7.9,Year:2008},{Id:2,Name:Monsters
Inc.,Rating:8.0,Year:2001}]";
var list2 = JsonConvert.DeserializeObject<List<Movie>>(json2);
foreach (var item in list2) Console.WriteLine(item.Name);

```

<i>Критерии за оценяване на изпитна тема № 12</i>	<i>Максимален брой точки</i>
1. Познава инструментите за разработка на софтуер в интегрирана среда за разработка.	20
2. Продуктивно използва интегрираната среда за разработка.	20
3. Използва външни библиотеки и пакети.	20
4. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 13: Разработка на софтуер*

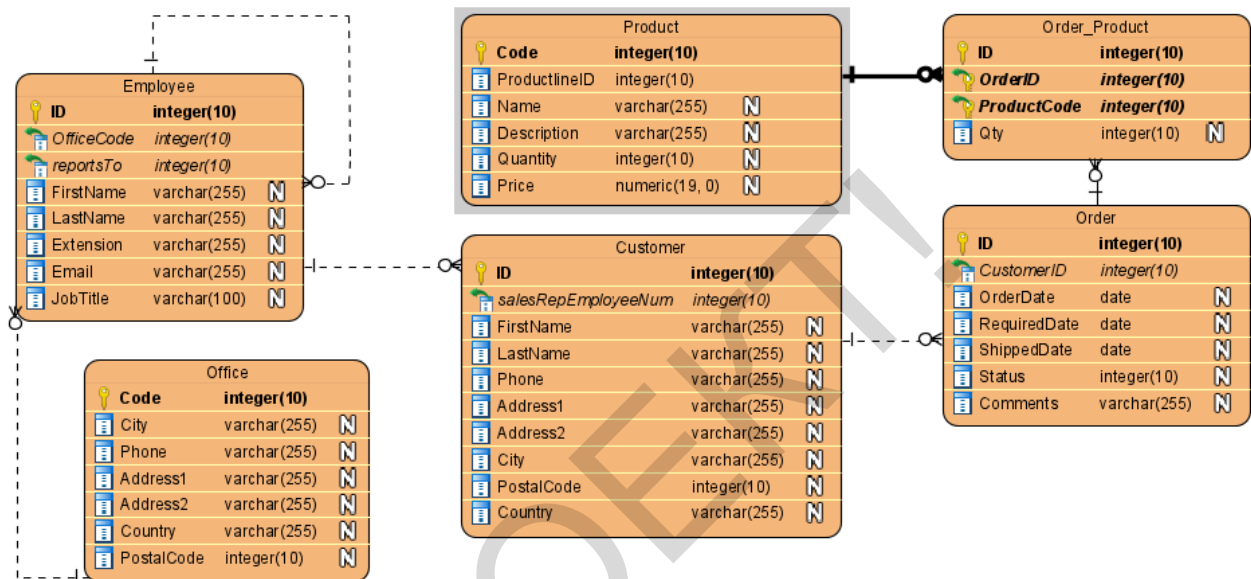


**План-тезис:** *Свързване на приложения с бази от данни. Използване на ORM система. Създаване на приложения с няколко потребителски интерфейса.*

**Примерна приложна задача:**

**Условие:**

Разполагате със следната схема. Създайте класове, които да ѝ съответстват и да са подходящи за работа с ORM система.



<b>Критерии за оценяване на изпитна тема № 13</b>	<b>Максимален брой точки</b>
1. Описва свързването на приложения с бази от данни.	20
2. Познава ORM технологиите.	20
3. Познава добрите практики при създаване на приложения с няколко потребителски интерфейса.	20
4. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

**Изпитна тема № 14: Операционни системи**

**План-тезис:** *Структура на компютърните системи и операционни системи. Процеси и памет. Команди и команден интерпретатор. Пакетни системи и инсталиране на софтуер в операционните системи.*

**Примерна приложна задача:**

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Напишете команди, които изпълняват следните задачи:

1. Изведете свободното дисково пространство в системата.
2. Изведете количеството свободна оперативна памет.
3. Намерете номерата (PID) на всички процеси свързани със systemd.

Фрагмент:

Команди
<pre>df -h free -m pgrep systemd</pre>

<b>Критерии за оценяване на изпитна тема № 14</b>	<b>Максимален брой точки</b>
1. Познава структура на компютърните системи и операционни системи.	12
2. Разбира работата на процесите и паметта в операционните системи. Разбира разликите между процес и програма.	12
3. Познава командния интерпретатор и знае да използва команди без графична среда в терминала.	18
4. Познава пакетни системи и начини за инсталиране на софтуер в операционните системи.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 15: Операционни системи*

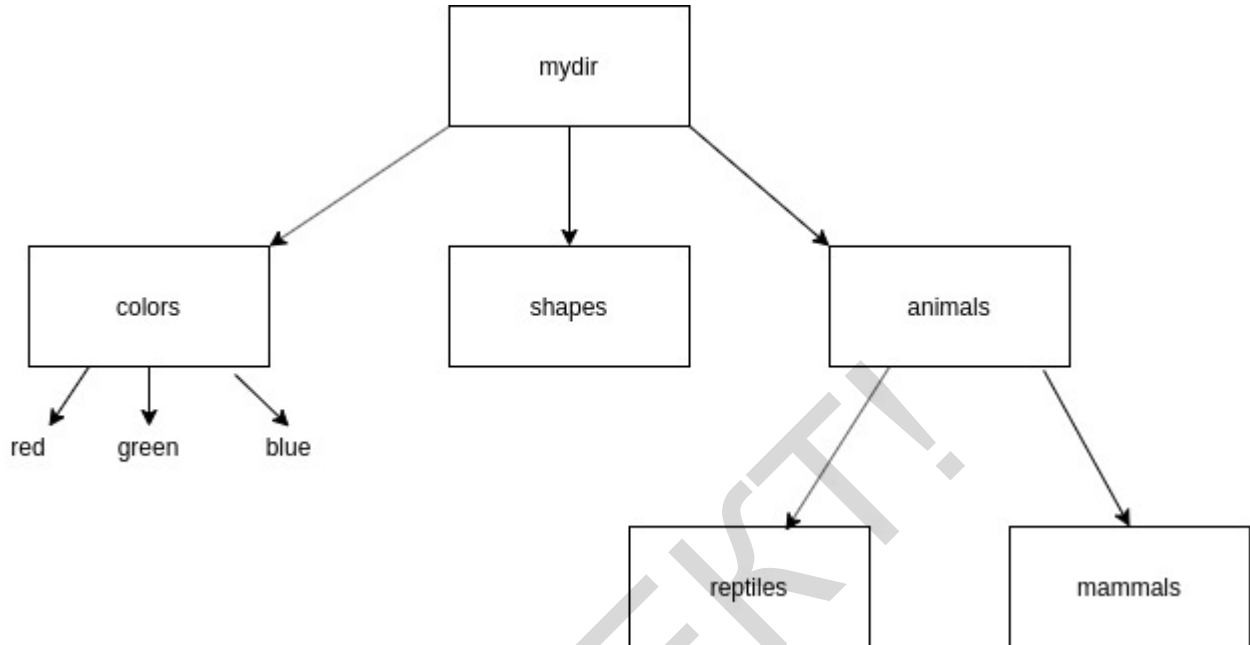
*План-тезис: Базови и мрежови услуги. Стартиране и спиране на услуги ръчно и по график. Файлови системи. Физическа и логическа организация на файловата система. Монтиране на файлова система, разделяне на дялове, конфигурация, форматиране, работа с файлове.*

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Създайте чрез команди необходимите директории и файлове от следната диаграма:



Файловете red, green и blue трябва да съдържат като съдържание шестнадесетичните кодове на всеки от цветовете. След създаване на всички файлове и директории архивирайте mydir и всички поддиректории и файлове.

Фрагмент:

Команди

```
mkdir mydir
cd mydir
mkdir colors shapes animals
cd animals
mkdir reptiles mammals
cd ../colors
touch red green blue
echo '#ff0000' > red
echo '#00ff00' > green
echo '#0000ff' > blue
cd ../../
zip -r mydir.zip mydir
```

*Критерии за оценяване на изпитна тема № 15*

*Максимален  
брой точки*

1. Познава основните базови и мрежови услуги. Описва стартиране и спиране на услуги по график.	12
2. Познава и различава различни видове файлови системи. Дефинира физическа и логическа организация на файловата система.	12
3. Описва монтиране на файлова система, разделяне на дялове, конфигурация и форматиране.	18
4. Познава принципите на работа с файлове в различни файлови системи.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

*Изпитна тема № 16: Математически основи на програмирането*

*План-тезис: Бройни системи. Преобразуване от една бройна система към друга. Операции в бройни системи. Статистика. Генерална съвкупност и извадка. Средна стойност, мода и медиана. Графични представяния на статистически данни - полигон, хистограма, кръгова диаграма. Свойства на функциите. Правоъгълна координатна система. Изобразяване на графика на функция. Системи линейни уравнения. Методи за решаване на системи линейни уравнения с повече неизвестни.*

Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

Условие:

Решете следната система линейни уравнения, чрез използването на избрана изучавана софтуерна среда:

$$\begin{cases} x_1 + 2x_2 + 5x_3 = -9 \\ x_1 - x_2 + 2x_3 = 3 \\ 3x_1 - 6x_2 - x_3 = 25 \end{cases}$$

Фрагмент:

solution.py

```
import math
import matplotlib.pyplot as plt
import numpy as np
```

```
a = np.array([[1,2,5], [1,-1,2], [3, -6, -1]])
b = np.array([-9,3,25])
x = np.linalg.solve(a, b)
print(x) // [ 2. -3. -1.]
```

<b>Критерии за оценяване на изпитна тема № 16</b>	<b>Максимален брой точки</b>
1. Познава бройни системи. Различава записи на числа в различни бройни системи. Умее да преобразува числа от една бройна система в друга.	12
2. Дефинира понятия от статистиката - генерална съвкупност и извадка, средна стойност, мода и медиана. Познава различни начини за графично представяне на статистически данни.	12
3. Разбира и прилага методи за решаването на СЛУ.	18
4. Познава функциите и техните свойства. Умее да чертае графики на математически функции.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

### *Изпитна тема № 17: Математически основи на програмирането*

*План-тезис: Вектор. Свойства на векторите. Връзка между вектори и масиви в програмирането. Множества. Операции с множества. Комбинаторика. Основни комбинаторни конфигурации - пермутации, комбинации и вариации. Елементи от теория на вероятностите. Събития, вероятност на събитие, условна вероятност. Пресмятане на вероятности.*

#### Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

#### Условие:

Да се напише програма, която генерира всички пермутации съставени от естествените числа от 1 до **n**.

#### Вход

От конзолата се прочита **1 цяло число** в интервала [1...7]

#### Изход

Да се отпечатаат елементите на всяка пермутация на отделни редове, разделени с интервал.

Примерен вход и изход

Примери:

Вход	Изход
3	1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1

Фрагмент:

```
Permutations.cs  
  
public static void Gen(int index)  
{  
    if (index >= elements.Length)  
        Console.WriteLine(string.Join(" ", perm));  
    else  
        for (int i = 0; i < elements.Length; i++)  
            if (!used[i])  
            {  
                used[i] = true;  
                perm[index] = elements[i];  
                Gen(i + 1);  
            }  
}
```

<i>Критерии за оценяване на изпитна тема № 17</i>	<i>Максимален брой точки</i>
1. Познава векторите и техните свойства. Обяснява връзката между векторите и масивите в програмирането.	12
2. Познава понятието множество и различни множества. Дефинира операции с множества.	12
3. Познава комбинаторни конфигурации. Различава пермутации, комбинации и вариации.	18
4. Описва алгоритмите за генериране на комбинаторни конфигурации. Умее да пресмята вероятности.	18
5. Решава приложната задача/казус.	40
<b>Общ брой точки:</b>	<b>100</b>

## Изпитна тема № 18: Конкурентно програмиране

**План-тезис:** Конкурентност. Изпълнение на програма. Процес. Блокиращи операции. Видове блокиращи операции. Нишка. Връзка между процес и нишка. Създаване на нишки. Управление на нишки. Синхронизация между нишки. Проблеми при работа с нишки.

### Примерна приложна задача:

По време на теоретичния изпит се предоставя непълен/неработещ/некоректен програмен фрагмент на приложната задача. Предоставеният фрагмент да се приведе в работещ вид.

### Условие:

Да се напише програма, която получава цяло число  $n$ . След това програмата да генерира масива с  $n$  на брой елементи от случайни числа в интервала  $[-100; 100]$ . Програмата трябва да разполага с две паралелни нишки, които пресмятат съответно сумата и произведението от елементите в масива. На всяка итерация от пресмятането трябва да се извежда текущия резултат.

Забележка: Понеже произведението може да надхвърли сравнително лесно стандартния диапазон на 64-битовите типове, използвайте подходящ клас (`BigInteger`) за големи числа.

### Фрагмент:

Calculator.cs

```
public class Calculator
{
    private int[] array;

    public Calculator(int[] Array)
    {
        this.array = Array;
    }

    public void Sum()
    {
        int sum = 0;
        for (int i = 0; i < array.Length; i++)
        {
            Console.WriteLine($"Sum of elements from 0 to {i} = {sum}");
            sum += array[i];
            Thread.Sleep(500);
        }
    }

    public void Product()
    {
        int product = 1;
```

```

    for (int i = 0; i < array.Length; i++)
    {
        Console.WriteLine($"Product of elements from 0 to {i} = {product}");
        product *= array[i];
        Thread.Sleep(500);
    }
}

```

Program.cs

```

public class Program
{
    static void Main(string[] args)
    {
        int arraySize = int.Parse(Console.ReadLine());

        int[] array = new int[arraySize];
        Random random = new Random();
        for (int i = 0; i < arraySize; i++)
        {
            array[i] = random.Next(-100, 100);
        }
        Console.WriteLine($"Array: {String.Join(", ", array)}");

        Calculator calculator = new Calculator(array);
        Thread sumThread = new Thread(calculator.Sum);
        Thread productThread = new Thread(calculator.Product);
        sumThread.Start();
        productThread.Start();
    }
}

```

<b>Критерии за оценяване на изпитна тема № 18</b>	<b>Максимален брой точки</b>
1. Дефинира понятието конкурентност. Различава видове конкурентност. Описва изпълнението на програма. Дефинира понятията процес, блокираща операция, нишка. Различава видове блокиращи операции.	12
2. Познава понятието нишка. Описва връзката между процес и нишка.	12
3. Знае начините за създаване и управление на нишки. Описва синхронизацията между нишки.	18
4. Познава различни проблеми при работа с нишки и описва техните решения.	18
5. Решава приложната задача/казус.	40



#### IV. УКАЗАНИЯ ЗА СЪДЪРЖАНИЕТО НА ИНДИВИДУАЛНИТЕ ЗАДАНИЯ

Индивидуалното задание съдържа пълното наименование на училището/обучаващата институция, празни редове за попълване трите имена на обучавания, квалификационната форма, началната дата и началния час на изпита, крайния срок на изпита – дата и час, темата на индивидуалното практическо задание и изискванията към крайния резултат от изпълнението на заданието. По решение на комисията могат да се дадат допълнителни указания, които да подпомогнат обучавания при изпълнение на индивидуалното практическо задание.

Индивидуалните задания се изготвят от комисията за провеждане и оценяване на изпита част по практика на професията и специалността в училището/обучаващата институция. Всеки обучаван изтегля индивидуалното си задание, в което веднага саморъчно написва трите си имена.

##### Примерно индивидуално практическо задание № 1:

##### 1. Указания (инструкции/изисквания) за изпълнение на индивидуалното задание:

- .....
- .....
- .....
- .....

##### 2. Критерии за оценяване

За всяко индивидуално задание комисията по провеждане и оценяване на изпита по практика на професията и специалността, назначена със заповед на директора на училището/ръководителя на обучаващата институция, разработва показатели по критериите, определени в таблицата. Посочва се максималният брой точки, които се поставят при пълно, вярно и точно изпълнение на показателя.

##### *Пример:*

<i>Критерии и показатели за оценяване</i>	<i>Максимален брой точки</i>	<i>Те-жест</i>
<b>1. Спазване на правилата за здравословни и безопасни условия на труд и опазване на околната среда</b>		<b>да/не</b>

1.1. Избира и използва правилно лични предпазни средства		
1.2. Правилно и по безопасен начин използва предметите и средствата на труда		
1.3. Разпознава опасни ситуации, които биха могли да възникнат в процеса на работа, дефинира и спазва предписания за своевременна реакция		
<i>Забележка: Критерий 1 няма количествено изражение, а качествено. Ако обучаваният по време на изпита създава опасна ситуация, застрашаваща собствения му живот или живота на други лица, изпитът се прекратява и на обучавания се поставя оценка <b>слаб (2)</b>.</i>		
<b>2. Ефективна организация на работното място</b>		<b>5</b>
2.1. Подреденост на инструменти, пособия и материали, осигуряваща удобство и точно спазване на технологията	2	
2.2. Целесъобразна употреба на материалите	2	
2.3. Работа с равномерен темп за определено време	1	
<b>3. Спазване изискванията на правилниците, наредбите и предписанията</b>		<b>5</b>
3.1. Обяснява работата си при спазване на йерархична подчиненост от други лица	3	
3.2. Спазва изискванията на правилниците, наредбите и предписанията, свързани с изпитното задание (материали, инструменти, лични предпазни средства)	2	
<b>4. Правилен подбор на детайли, материали и инструменти съобразно конкретното задание</b>		<b>20</b>
4.1. Преценява типа и вида на необходимите материали, детайли и инструменти според изпитното задание	10	
4.2. Правилно подбира количеството и качеството на необходимите материали, детайли и инструменти	10	
<b>5. Спазване на технологичната последователност на операциите според практическото изпитно задание</b>		<b>20</b>
5.1. Самостоятелно определя технологичната последователност на операциите	10	
5.2. Спазва технологичната последователност на операциите в процеса на работа	10	
<b>6. Качество на изпълнението на индивидуалното практическо задание</b>		<b>50</b>
6.1. Всяка завършена операция съответства на изискванията на съответната технология	20	
6.2. Крайното изделие съответства на зададените технически параметри	20	
6.3. Изпълнява задачата в поставения срок	10	
<b>Общ брой точки:</b>		<b>100</b>

## V. СИСТЕМА ЗА ОЦЕНЯВАНЕ

Оценяването на резултатите от държавния изпит за придобиване на втора степен на професионална квалификация по специалността код **481010** „Програмно осигуряване“, професия код **4810101** „Програмист“ е в точки, както следва:

- част по теория на професията - максимално 100 точки;
- част по практика на професията - максимално 100 точки.

Всяка част от държавния изпит е успешно положена при постигане на петдесет на сто от максималния брой точки.

Формирането на окончателната оценка от изпита е в съотношение - 40 процента частта по теория на професията и 60 процента частта по практика на професията от общия брой точки.

Окончателната оценка в брой точки се формира след успешното полагане на всяка част от изпита и се изчислява, както следва:

Окончателната оценка в брой точки е равна на  $0,4 \times$  получения брой точки от частта по теория на професията +  $0,6 \times$  получения брой точки от частта по практика на професията.

Окончателната оценка от брой точки се превръща в цифрова оценка с точност до 0,01 по формулата:

**Цифрова оценка = окончателната оценка в брой точки  $\times$  0,06.**

Окончателната оценка от държавния изпит за придобиване на квалификация по професията е с количествен и качествен показател, с точност до 0,01 и се определя, както следва:

- а) за количествен показател от 2,00 до 2,99 се определя качествен показател слаб;
- б) за количествен показател от 3,00 до 3,49 се определя качествен показател среден;
- в) за количествен показател от 3,50 до 4,49 се определя качествен показател добър;
- г) за количествен показател от 4,50 до 5,49 се определя качествен показател много добър;
- д) за количествен показател от 5,50 до 6,00 се определя качествен показател отличен.

## VI. ПРЕПОРЪЧИТЕЛНА ЛИТЕРАТУРА

1. Наков С., Колев В. и колектив, Въведение в със C#, София, 2015, ISBN 978-954-400-527-6, <http://www.introprogramming.info/intro-csharp-book/>
2. Наков С. и колектив, Въведение в с Java, София, 2008, ISBN 978-954-400-055-4, <http://www.introprogramming.info/intro-java-book/>

3. Abelson H., Sussman G., Structure and Interpretation of Computer Programs, MIT Press, London, 1996
4. McLaughlin, B. D.; Pollice, G. & West, D. (2007), Head first object-oriented analysis and design - a brain-friendly guide to OOA&D., O'Reilly
5. Gamma, e. a. (1994), Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley Professional
6. Weisfeld, M. (2013), The Object-Oriented Thought Process (Developer's Library) 4th Edition, Addison-Wesley Professional
7. Beaulieu A., Learning SQL: Master SQL Fundamentals, O'Reilly Media; 2nd edition (2009)
8. Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring 2nd edition, John F. Dooley, Apress, 2017, ISBN 978-1484231524
9. Extreme Programming Explained: Embrace Change, 2nd Edition, Kent Beck, Addison-Wesley, 2005, ISBN 978-0321278654
10. Modern Operating Systems (4th Edition), Andrew S. Tanenbaum, Pearson, 2014, ISBN 978-0133591620
11. Operating Systems Concepts, Abraham Silberschatz, Greg Gagne, Peter Baer Galvin, Wiley, 2012, ISBN 978-1118063330
12. Operating Systems: Design and Implementation, 3rd edition, Albert S. Woodhull, Andrew S. Tanenbaum, Pearson, 2006, ISBN 978-0136373315
13. Exploring Arduino: Tools and Techniques for Engineering Wizardry, Jeremy Blum, Wiley, 2013, ISBN 978-1118549360
14. Programming Arduino: Getting Started with Sketches, Second Edition, Simon Monk, McGraw-Hill Education, 2016, ISBN 978-1259641633
15. Make: Arduino Bots and Gadgets: Six Embedded Projects with Open Source Hardware and Software, Tero Karvinen, Kimmo Karvinen, Maker Media, 2011, ISBN 978-1449389710
16. Hal Abelson's, Jerry Sussman's and Julie Sussman's Structure and Interpretation of Computer Programs, MIT Press, 1984; ISBN 0-262-01077-1, <https://mitpress.mit.edu/sicp/>
17. Miran Lipovača, Learn You a Haskell, No Starch Press, ISBN-13: 978-1-59327-283-8, <http://learnyouahaskell.com>
18. Paul Chiusano, Rúnar Bjarnason, Functional Programming in Principles with Scala, Manning Publications, ISBN-13: 978-1617290657, <https://www.manning.com/books/functional-programming-in-scala>
19. Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring 2nd edition, John F. Dooley, Apress, 2017, ISBN 978-1484231524
20. Refactoring: Improving the Design of Existing Code (2nd Edition), Martin Fowler, Addison-Wesley, 2018, ISBN 978-0134757599
21. The Pragmatic Programmer, Andy Hunt, Dave Thomas, Addison-Wesley, 1999, ISBN 978-0201616224
22. Clean Code, Robert Martin, Prentice Hall, 2008, ISBN 978-0132350884
23. Code Complete: A Practical Handbook of Software Construction, Second Edition, Steve McConnell, Microsoft Press, 2004, 978-0735619678

### **Електронни информационни източници:**

1. Портал за е-Обучение в специалност „Приложен програмист“ <https://it-kariera.mon.bg/e-learning>
2. Програмиране 101 към Хак България <https://github.com/HackBulgaria/Programming101-Python-2016>

## **VII. АВТОРСКИ КОЛЕКТИВ**

Програмата е разработена, обсъдена и оформена от експертна група към Национална програма „Обучение за ИТ кариера“ към МОН в състав:

1. доц. д-р Димитър Минчев, Бургаски свободен университет, Бургас
2. доц. д-р Ивайло Старибратов - ПУ „Паисий Хилендарски“
3. Петър Петров – ПГЕЕ „Константин Фотинов“, Бургас
4. Росен Вълчев, МГ „Акад. Кирил Попов“, Пловдив
5. инж. Хриси Плачкова, МГ „Акад. Кирил Попов“, Пловдив

ПРОЕКТ!



**б) Примерно индивидуално практическо задание**

.....  
(пълно наименование на училището/обучаващата институция)

**ДЪРЖАВЕН ИЗПИТ - ЧАСТ ПО ПРАКТИКА НА ПРОФЕСИЯТА И  
СПЕЦИАЛНОСТТА,**

**ЗА ПРИДОБИВАНЕ НА ВТОРА СТЕПЕН НА ПРОФЕСИОНАЛНА  
КВАЛИФИКАЦИЯ**

**по професия код 4810101 „Програмист“  
специалност код 481010 „Програмно осигуряване“**

**Индивидуално практическо задание № .....**

На ученика/обучавания .....

(трите имена на ученика/обучавания)

от .....клас/курс, начална дата на изпита: ..... начален час: .....

крайна дата на изпита: ..... час на приключване на изпита: .....

1. Да се .....

(вписва се темата на практическото задание)

2. Указания (инструкции/изисквания) за изпълнение на практическото задание:

.....  
.....  
.....  
.....

УЧЕНИК/ОБУЧАВАН: .....

(име, фамилия)

(подпис)

Председател на изпитната комисия: .....

(име, фамилия)

(подпис)

Директор/ръководител на обучаващата институция: .....

(име, фамилия) (подпис)  
(печат на училището/обучаващата институция)

**в) Примерно указание за разработване на писмен тест**

- **примерно указание за работа за учениците/курсистите и примерни тестови задачи с еталон за оценяване и ключ на верните отговори**

Указание за работа

Уважаеми ученици/курсисти,

Вие получавате тест, който съдържа ... задачи с различна трудност с максимален брой точки – 100. За всеки Ваш отговор ще получите определен брой точки, показан в долния десен ъгъл след всяка задача.

Целта на теста е да се установи равнището на усвоените от Вас знания и умения, задължителни за усвояване и контрол за придобиване на втора степен на професионална квалификация по професия „Програмист“, специалност „Програмно осигуряване“.

Отбелязването на верния според Вас отговор при задачите с избран отговор е чрез знак  $\times$ , а за другите типове задачи начина на отговор е описан в задачата.

При отбелязване на отговор, който искате да промените, оградете в кръгче грешното отбелязване и се подпишете пред него.

Някои задачи изискват не само познаване на учебното съдържание, но и логическо мислене, затова четете внимателно условията на задачите преди да посочите някой отговор.

Не отделяйте много време на въпрос, който Ви се струва труден, върнете се на него по-късно, ако Ви остане време.

Тестът е с продължителност ..... астрономически часа.

**ЖЕЛАЕМ ВИ УСПЕХ !**

- **разработване на тест**

Броят и равнището на тестовите задачи по всеки критерий се определят съобразно равнището, на което трябва да бъде усвоено съответното учебно съдържание, като общият брой задачи по всеки критерий трябва да носи максималния брой точки.

**1. Таксономия на Блум – равнища и примерни глаголи**

Равнище	Характеристика	Глаголи
<b>I. Знание 0 - 2 точки</b>	Възпроизвеждане и разпознаване на информация за понятия, факти, дефиниции	Дефинира, описва, посочва, изброява, очертава, възпроизвежда, формулира, схематизира



<b>II. Разбиране 0 - 4 точки</b>	Извличане на съществен смисъл от изучаваната материя. Интерпретация и трансформиране на информацията с цел нейното структуриране.	Преобразува, различава, обяснява, обобщава, преразказва, решава, дава пример за..., сравнява
<b>III. Приложения 0 - 6 точки</b>	Пренос на нови знания и умения при решаване на проблемна или аварийна ситуация. Способност за използване на усвоената информация и формираните умения	Изчислява, демонстрира, открива, модифицира, разработва, свързва, доказва

## 2. Примерна матрица на писмен тест по изпитна тема № ....

Разработка се от комисията за подготовка и оценяване на изпита - част по теория на професията, като към таблицата за критерии за оценка по всяка тема се разписват графи 3, 4 и 5.

	<b>М а кс и м а ле н Б р о й т о ч к и</b>	<b>Брой тестови задачи по равнища</b>		
		<b>I</b>	<b>II</b>	<b>III</b>
		<b>З н а н и е 0 - 2 т .</b>	<b>Р а з б и р а н е 0 - 4 т .</b>	<b>П р и л о ж е н и е 0 - 6 т .</b>
<b>Критерии за оценяване на изпитна тема № ....</b>				
<i>1</i>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
1. ....				
2. ....				
3. ....				
4. ....				
.....				
<b>Общ брой задачи:</b>				
<b>Общ брой точки:</b>	<b>100</b>			
<b>При оценка на резултатите от теста максимален брой точки се поставя при отговор, съвпадащ с ключа за оценяване, както следва:</b>				
<ul style="list-style-type: none"> <li>• 2 точки за тестовите задачи от равнище „Знание“</li> <li>• 4 точки за тестовите задачи от равнище „Разбиране“</li> </ul>				

### 3. Препоръчителни тестови въпроси и задачи според типа на отговора:

- **1-ва група: въпроси и задачи със свободен отговор;**
  - Въпроси и задачи за свободно съчинение;
  - Въпроси и задачи за тълкуване;
- **2-ра група: въпроси и задачи за допълване (с полуоткрит отговор);**
  - Въпроси и задачи за допълване на дума, или фраза или елемент от чертеж/схема;
  - Въпроси и задачи за заместване;
- **3-та група: въпроси и задачи с избран отговор**
  - Задачи с един или повече верни отговори;
  - Въпроси за избор между вярно и грешно

### 4. Примерни тестови задачи

#### 4.1. *Примерна тестова задача от равнище „Знание“*

Посочете кое от изброените влакна има най-голяма хигроскопичност:

- а) памучно
- б) вълнено
- в) ленено
- г) естествена коприна
- д) изкуствена коприна

**Еталон на верния отговор:** б)

**Ключ за оценяване:**

Отговор б) – 2 точки

При посочени повече от един отговор – 0 точки

Всички останали отговори – 0 точки

#### 4.2. *Примерна тестова задача от равнище „Разбиране“*

Определете вярното твърдение:

- а) чрез решенето се постигат високи качествени показатели по отношение на гладкост, равномерност, здравина;

б) чрез решенето се намалява големината на отделните снопчета от влакна и тяхното разпухване по механичен начин;

в) чрез решенето на влакнестата маса получава нов вид, различаващ се от първоначалния по честота, форма и еднородност.

**Еталон на верния отговор:** а)

**Ключ за оценяване:**

Отговор а) - 4 точки;

При посочени повече от един отговор - 0 точки т;

Всички останали отговори - 0 точки;

**4.3.    *Примерна тестова задача от равнище „Приложение“:***

Попълнете пропуснатите технологични процеси при камгарната система на предене

**Еталон на верния отговор и ключ за оценяване:**