МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ ПО ИНФОРМАТИКА

20 май 2024 г.

ПРОФИЛИРАНА ПОДГОТОВКА

ВАРИАНТ 1

Задача от 1. до 16. Ключ с верните отговори

Въпрос №	Верен отговор	Брой точки
1.	А	1
2.	В	1
3.	Б	1
4.	Г	1
5.	Б	1
6.	В	1
7.	А	1
8.	В	1
9.	В	1
10.	Б	1
11.	Г	1
12.	В	1
13.	Б	1
14.	Г	1
15.	В	1
16.	Г	1

Задача 17. – 2 точки

А) `6`

Б) `(b-a)/step+1`

Задача 18. – 2 точки

```
242482
```

```
9131
```

Задача 19. – 2 точки

```
(1)     3 -8 -3 или
   3.0 -8.0 -3.0
```

```
(2)     6 3 или

   6.0 3.0

(3)     [-16, -3, -1, 0, 9, 100]
```

(4) NaN или друг отговор в смисъл, че „не може да се намери квадратен корен от отрицателно число“ или да се приложи Math.Sqrt() върху отрицателен аргумент.

Задача 20. – 3 точки

```
(1) ON d.department_id = e.department_id
(2) WHERE
(3) ORDER BY
```

Задача 21. – 3 точки

```
(1) private string make (за C#) / private String make (за Java)
(2) public void Accelerate (int speed)   (за C#)
      public void accelerate (int speed)   (за Java)
(3) public int GetSpeed () (за C#) / public int getSpeed () (за
Java)
```

Задача 22. – 3 точки

```
(1) points
(2) sortedPoints
(3) i + 1
```

Задача 23. – 3 точки

```
ред 2 for (int i = 0; i < arr.Length; i++) или arr.Length (за C#)
      for (int i = 0; i < arr.length; i++) или arr.length (за Java)
ред 4 return i+1; или i+1
ред 5 return 0; или 0
```

Задача 24. – 3 точки

```
1  int [] arr = {25, -3, 10, -1, 40, -12};

2  int minEl = arr[0];
```

Признава се и задаването на който и да е друг индекс на елемент от масива. Не се признава за вярна редакция, ако `arr` се замени с конкретна стойност

```
3  for (int i = 1 ; i < arr.Length; i++) (за C#)

   for (int i = 1 ; i < arr.length; i++) (за Java)
```

Задача 25. – 3 точки

2. chocolate
4. coffee
6. water

Задача 26 – 15 точки

Примерно решение

C#

```
using System;

namespace DZI
{
    class Zad26
    {
        static void Main(string[] args)
        {
            int a = int.Parse(Console.ReadLine());
            int b = int.Parse(Console.ReadLine());
            int[] count = new int[10];
            for (int i = a; i <= b; i++)
            {
                int x = i;
                while (x != 0)
                {
                    int c = x % 10;
                    count[c]++;
                    x = x / 10;
                }
            }
            int max = count[0];
            int digit = 0;
            for (int i = 1; i < 10; i++)
            {
                if (count[i] > max)
                {
                    max = count[i];
                    digit = i;
                }
            }
            Console.WriteLine($"Digit {digit} - {max} times");
        }
    }
}
```

Java

```
package DZI;

import java.util.Scanner;

public class Zad26 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int[] count = new int[10];

        for (int i = a; i <= b; i++) {
            int x = i;
            while (x != 0) {
                int c = x % 10;
                count[c]++;
                x = x / 10;
            }
        }

        int max = count[0];
        int digit = 0;
        for (int i = 1; i < 10; i++) {
            if (count[i] > max) {
                max = count[i];
                digit = i;
            }
        }
        System.out.printf("Digit %d - %d times%n", digit, max);
    }
}
```

Задача 27 – 20 точки

Примерно решение

```
CREATE DATABASE Geo;
USE Geo;
--- A)
CREATE TABLE Mountains (
ID INT NOT NULL IDENTITY PRIMARY KEY,
```

```
MountainName NVARCHAR(20) NOT NULL,
CountryCode CHAR(3) NOT NULL,
Country NVARCHAR(20));

--- Б)
CREATE TABLE Peaks (
ID INT NOT NULL IDENTITY PRIMARY KEY,
PeakName NVARCHAR(20) NOT NULL,
Elevation INT NOT NULL,
MountainId INT NOT NULL REFERENCES Mountains(ID),
CHECK (Elevation > 0));

--- В)
INSERT INTO Mountains (MountainName,CountryCode, Country) VALUES
('Рила','BUL','България'),
('Пирин','BUL','България'),
('Стара планина','BUL','България'),
('Анди','ARG','Аржентина'),
('Анди','CHL','Чили'),
('Хималаи','NPL','Непал'),
('Алпи','SUI','Швейцария'),
('Алпи','ITA','Италия'),
('Алпи','AUT','Австрия'),
('Алпи','FRA','Франция'),
('Елбрус','RUS','Русия'),
('Елбрус','GEO','Грузия');

--- Г)
INSERT INTO Peaks VALUES
('Аконкагуа',6962,4),
('Ботев',2376,3),
('Мусала',2925,1),
('Еверест',8849,6),
('Вихрен',2914,2),
('Мальовица',2729,1),
('Монблан',4809,10),
('Матерхорн',4478,8),
('Дюфур',4634,7),
('Елбрус',5642,11),
('Ком',2015,3),
('Манаслу',2729,6),
('Дено',2790,1);

--- Д)
UPDATE Peaks
SET Elevation = 2016
WHERE PeakName = 'Ком';
```

```
--- Е)
SELECT AVG(Elevation) FROM Peaks
WHERE MountainId = 1;

--- Ж)
SELECT COUNT(ID) FROM Peaks
WHERE Elevation BETWEEN 5000 AND 9000;

--- З)
SELECT p.PeakName, p.Elevation, m.MountainName, m.Country FROM Peaks AS p
INNER JOIN Mountains AS m ON p.MountainId=m.ID
WHERE Elevation > 2900
ORDER BY Country, Elevation DESC;

--- И)
SELECT Country, COUNT(*) AS 'Count' FROM Mountains
GROUP BY Country
ORDER BY COUNT(*) DESC, Country;
```

Задача 28 – 25 точки

Примерно решение

C#

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;

namespace Zad28
{
    class Zad28
    {
        static void Main(string[] args)
        {
            using(StreamReader reader = new StreamReader("data.txt"))
            {
                KinderGarden kinderGarden = new KinderGarden();
                string cmd;
                while ((cmd = reader.ReadLine()) != "END")
                {
                    string[] cmdArgs = cmd.Split();
                    Kid kid;
                    switch (cmdArgs[0])
                    {
                        case "enrollment":
                            try
                            {
```

```
                                kid = new Kid(cmdArgs[1], cmdArgs[2],
cmdArgs[3], int.Parse(cmdArgs[4]), cmdArgs[5], cmdArgs[6]);
                                kinderGarden.EnrollKid(kid);
                            }
                            catch (Exception e)
                            {
                                Console.WriteLine(e.Message);
                            }
                            break;
                        case "unsubscribe":
                            kinderGarden.ReleaseKid(cmdArgs[1]);
                            break;
                        case "information":
                            kinderGarden.GroupInfo(cmdArgs[1]);
                            break;
                        default:
                            Console.WriteLine($"{cmdArgs[0]} - invalid
command");
                            break;
                    }
                }
                Console.WriteLine("Have a nice day!");
            }
        }
    }

class Person
    {
        private string id;
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string ID
        {
            get { return id; }
            set
            {
                if (value.Length != 10)
                {
                    throw new Exception($"{FirstName} {LastName} - invalid
identifier!");
                }
                id = value;
            }
        }
        public Person (string firstName, string lastName, string id)
        {
            FirstName = firstName;
            LastName = lastName;
            ID = id;
        }
    }

class Kid : Person
    {
        private int age;
```

```
        public int Age
        {
            get { return age; }
            set
            {
                if (value < 3 || value > 6)
                {
                    throw new Exception($"The child {FirstName} {LastName} age
is invalid - {value}");
                }
                age = value;
            }
        }
        public string Group { get; set; }
        public string ParentLastName { get; set; }
        public string ParentGSM { get; set; }
        public Kid(string firstName, string lastName, string id, int age, string
parentLastName, string parentGSM)
            : base(firstName, lastName, id)
        {
            Age = age;
            switch (age)
            {
                case 3:
                    Group = "I";
                    break;
                case 4:
                    Group = "II";
                    break;
                case 5:
                    Group = "III";
                    break;
                case 6:
                    Group = "IV";
                    break;
            }
            ParentLastName = parentLastName;
            ParentGSM = parentGSM;
        }
        public override string ToString()
        {
            return $"{FirstName} {LastName}, {Age}г., {ParentGSM}
({ParentLastName})";
        }

    }

class KinderGarden
    {
        private List<Kid> kidList;
        public KinderGarden()
        {
            kidList = new List<Kid>();
        }
        public void EnrollKid(Kid kid)
```

```
        {
            kidList.Add(kid);
            Console.WriteLine($"The child {kid.FirstName} {kid.LastName} is
enrolled.");
        }

        public void ReleaseKid(string id)
        {
            Kid kid = kidList.Find(k => k.ID == id);
            if (kid != null)
            {
                kidList.Remove(kid);
                Console.WriteLine($"The child {kid.FirstName} {kid.LastName} has
been unsubscribed.");
            }
            else Console.WriteLine($"Unsubscribe failed - invalid identifier
{id}.");
        }
        public void GroupInfo(string group)
        {
            int count = kidList.Count(k => k.Group == group);
            Console.WriteLine($"{group} group - {count} children");
            foreach (var kid in kidList.OrderBy(k => k.FirstName).ThenBy(k =>
k.LastName))
            {
                if (kid.Group == group)
                {
                    Console.WriteLine(kid);
                }
            }
        }

    }
}
```

Java

```
package dzi;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.*;

public class Zad28 {
    public static void main(String[] args) throws FileNotFoundException {
        try(Scanner reader = new Scanner(new FileInputStream("data.txt"))){
            KinderGarden kinderGarden = new KinderGarden();
            String cmd;
            while(reader.hasNext()){
                cmd = reader.nextLine();
                if(cmd.equals("END")) {
                    break;
                }
                String[] cmdArg = cmd.split(" ");
```

```
                switch (cmdArg[0]){
                    case "enrollment":
                        try{
                            Kid kid = new Kid(cmdArg[1], cmdArg[2], cmdArg[3],
Integer.parseInt(cmdArg[4]), cmdArg[5], cmdArg[6]);
                            kinderGarden.enrollKid(kid);
                        }
                        catch (Exception e){
                            System.out.println(e.getMessage());
                        }
                        break;
                    case "unsubscribe":
                        kinderGarden.releaseKid(cmdArg[1]);
                        break;
                    case "information":
                        kinderGarden.groupInfo(cmdArg[1]);
                        break;
                    default:
                        System.out.println(String.format("%s - invalid command
", cmdArg[0]));
                        break;
                }
            }
            System.out.println("Have a nice day!");
        }
    }
}

public class Person {

    private String firstName;
    private String lastName;
    private String id;

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getId() {
        return id;
    }
```

```
    public void setId(String id) throws Exception {
        if (id.length() != 10) {
            throw new Exception(String.format("%s %s - invalid identifier!",
getFirstName(), getLastName()));
        } else {
            this.id = id;
        }
    }

    public Person(String firstName, String lastName, String id) throws Exception
{
        this.firstName = firstName;
        this.lastName = lastName;
        this.setId(id);
    }
}

public class Kid extends Person {

    private int age;
    private String group;
    private String parentLastName;
    private String parentGSM;

    public int getAge() {
        return age;
    }

    public void setAge(int age) throws Exception {
        if (3 <= age && age <= 6) {
            this.age = age;
        } else {
            throw new Exception(String.format("The child %s %s age is invalid -
%d", getFirstName(), getLastName(), age));
        }
    }

    public String getGroup() {
        return group;
    }

    public void setGroup() {
        switch (getAge()) {
            case 3:
                this.group = "I";
                break;
            case 4:
                this.group = "II";
                break;
            case 5:
                this.group = "III";
                break;
            case 6:
                this.group = "IV";
                break;
```

```
        }
    }

    public String getParentLastName() {
        return parentLastName;
    }

    public void setParentLastName(String parentLastName) {
        this.parentLastName = parentLastName;
    }

    public String getParentGSM() {
        return parentGSM;
    }

    public void setParentGSM(String parentGSM) {
        this.parentGSM = parentGSM;
    }

    public Kid(String firstName, String lastName, String id, int age, String
parentLastName, String parentGSM) throws Exception {
        super(firstName, lastName, id);
        this.setAge(age);
        this.parentLastName = parentLastName;
        this.parentGSM = parentGSM;
        this.setGroup();
    }

    @Override
    public String toString() {
        return String.format("%s %s, %d, %s (%s)", getFirstName(),
getLastName(), getAge(), getParentGSM(), getParentLastName());
    }

}

public class KinderGarden {

    private List<Kid> kidList;

    public KinderGarden() {
        this.kidList = new ArrayList<>();
    }

    public void enrollKid(Kid kid) {
        kidList.add(kid);
        System.out.println(String.format("The child %s %s is enrolled.",
kid.getFirstName(), kid.getLastName()));
    }

    public void releaseKid(String id) {
        int i = 0;
        while ( i < kidList.size() && !kidList.get(i).getId().equals(id)) {
            i++;
        }
```

```
        if (i == kidList.size()) {
            System.out.println("Unsubscribe failed - invalid identifier " + id +
".");
        } else {
            System.out.println(String.format("The child %s %s has been
unsubscribed.", kidList.get(i).getFirstName(), kidList.get(i).getLastName()));
            kidList.remove(i);
        }
    }
    Comparator<Kid> compareByName = new Comparator<Kid>() {
        @Override
        public int compare(Kid a, Kid b) {
            if (a.getFirstName().equals(b.getFirstName())) {
                return a.getLastName().compareTo(b.getLastName());
            } else {
                return a.getFirstName().compareTo(b.getFirstName());
            }
        }
    };

    public void groupInfo(String group) {
        int cnt = 0;
        for (Kid el : kidList) {
            if (el.getGroup().equals(group)) {
                cnt++;
            }
        }
        Collections.sort(kidList, compareByName);
        System.out.println(String.format("%s group - %d children", group, cnt));
        for (Kid el : kidList) {
            if (el.getGroup().equals(group)) {
                System.out.println(el.toString());
            }
        }
    }
}
```