|  |  |
| --- | --- |
|  | Опросник будущего стажера  по разработки на Java |

На заполнение данного опросника у вас **1 день**.

**Ваши имя и фамилия: Гулай Станислав Леонидович**

Оцените свои знания по следующей шкале:

* 0 – Знания отсутствуют
* 1 – Начальные знания
* 2 – Знания среднего уровня
* 3 – Знания выше среднего или наличие существенного практического опыта

**ПО, языки, фреймворки и библиотеки**

|  |  |
| --- | --- |
| Предметная область | Знания |
| Java | 1 |
| C | 0 |
| C++ | 0 |
| C# | 0 |
| Python | 1 |
| intellij idea | 1 |
| Git | 1 |
| Confluence | 0 |
| Jira | 0 |
| Spring Data | 1 |
| Spring MVC | 1 |
| Spring Security | 1 |
| Spring Boot | 1 |
| Spring Cloud | 1 |
| Jenkins | 0 |
| MapStruct | 0 |
| Liquibase | 0 |
|  |  |

**Операционные системы**

|  |  |
| --- | --- |
| Предметная область | Знания |
| Windows | 2 |
| Linux | 1 |
| **SQL** | |
| Предметная область | Знания |
| SQL (Select, Join, Having, Group by) | 1 |
| SQL (Update, Insert, Delete, Trancate) | 1 |
|  |  |

**Другое\***

|  |  |
| --- | --- |
| Предметная область | Знания |
| CMD\Bat\Power Shell\bash | 1 |
| Регулярные выражения | 1 |
| Английский язык | 2 |
| Maven | 1 |
| Gradle | 1 |
| Docker | 1 |
|  |  |

\* - Если вы обладаете навыками, не включенными в таблицу, добавьте их самостоятельно.

**Задание 1.**

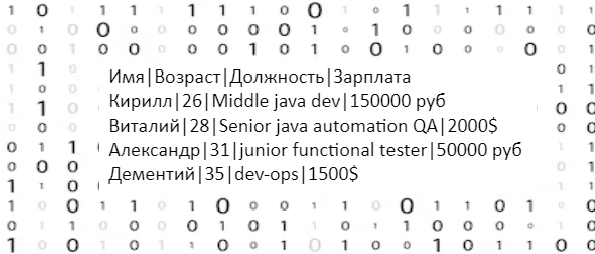
Расскажите о себе.

Почему хотите заниматься разработкой, чего ожидаете от нового места работы:

Работал в сфере энергетического машиностроения более десяти лет и параллельно получил степень кандидата технических наук в 2022 году. Во время обучения, где программирование было существенной частью диссертации понял, что хочу участвовать разработке программных продуктов и решил перейти в программирование.  
  
Для получения знаний и практики прошел обучение. Продолжаю постоянно самообучаться, читая туториалы, документацию, параллельно отрабатывая полученные знания. Сейчас нахожусь в поисках компании или проекта, где смогу принести максимальную пользу.

**Задание 2.**

Дана таблица:



Данная таблица представлена в формате

**List<Map<String,String>>**, где каждый элемент list - строка, key в map - название столбца, value в map - значение ячейки

Задача:

- вывести имена всех сотрудников, младше 30.

- вывести имена всех сотрудников, получающих зарплату в рублях.

- вывести средний возраст всех сотрудников.

\* **решение на Java**

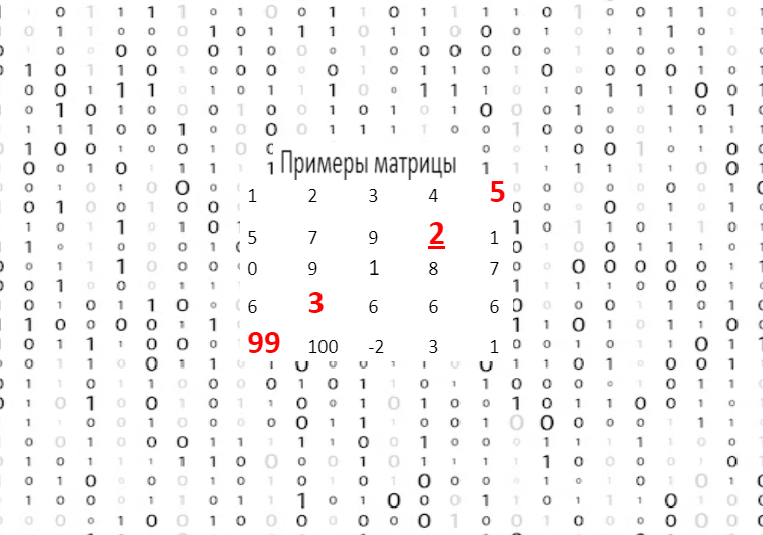
\* будет плюсом использование stream API

**Решение:**

enum Сolumn {  
 *name*, *age*, *position*, *salary*}  
  
class Employee {  
 Map<String, String> employee;  
  
 public Employee(String name, String ages, String position, String salary) {  
 employee = new HashMap<>();  
 employee.put(Сolumn.*name*.name(), name);  
 employee.put(Сolumn.*age*.name(), ages);  
 employee.put(Сolumn.*position*.name(), position);  
 employee.put(Сolumn.*salary*.name(), salary);  
  
 }  
  
  
 public String getName() {  
 return employee.get(Сolumn.*name*.name());  
 }  
  
 public String getAge() {  
 return employee.get(Сolumn.*age*.name());  
 }  
  
 public String getPosition() {  
 return employee.get(Сolumn.*position*.name());  
 }  
  
 public String getSalary() {  
 return employee.get(Сolumn.*salary*.name());  
 }  
  
 public String toString() {  
 return "Employee{" +  
 Сolumn.*name*.name() + ": " + this.getName() + "; " +  
 Сolumn.*age*.name() + ": " + this.getAge() + "; " +  
 Сolumn.*position*.name() + ": " + this.getPosition() + "; " +  
 Сolumn.*salary*.name() + ": " + this.getSalary() +  
 "}";  
 }  
  
}  
  
  
class Employees {  
  
 private final List<Employee> employees;  
  
 private void initEmployees() {  
 employees.add(new Employee("Кирилл", "26",  
 "Middle java dev", "150000 руб."));  
 employees.add(new Employee("Виталий", "28",  
 "Senior java automation QA", "2000$"));  
 employees.add(new Employee("Александр", "31",  
 "junior functional tester", "50000 руб"));  
 employees.add(new Employee("Дементий", "35",  
 "dev-ops", "1500$"));  
 }  
  
 public Employees() {  
 employees = new ArrayList<>();  
 initEmployees();  
 }  
  
 public List<Employee> getEmployees() {  
 return employees;  
 }  
  
 public List<String> getNamesYoungerThanAges(int maxAge) {  
 return getEmployees()  
 .stream()  
 .filter(e -> Integer.*parseInt*(e.getAge()) < maxAge)  
 .map(Employee::getName)  
 .toList();  
 }  
  
 public List<String> getNamesGetSalaryByRubles() {  
 return getEmployees()  
 .stream()  
 .filter(e -> e.getSalary().contains("руб"))  
 .map(Employee::getName)  
 .toList();  
 }  
  
 public double averageAgesAllEmployees() {  
 int sum = getEmployees()  
 .stream()  
 .map(Employee::getAge)  
 .map(Integer::*valueOf*)  
 .mapToInt(Integer::intValue)  
 .sum();  
 return (double) sum / getEmployees().size();  
 }  
  
 @Override  
 public String toString() {  
 return "Employees" + employees;  
 }  
  
}  
  
  
public class Main {  
 public static void main(String[] args) {  
 Employees employees = new Employees();  
 System.*out*.println(employees.getNamesYoungerThanAges(30));  
 System.*out*.println(employees.getNamesGetSalaryByRubles());  
 System.*out*.println(employees.averageAgesAllEmployees());  
 }  
}

**Задание 3.**

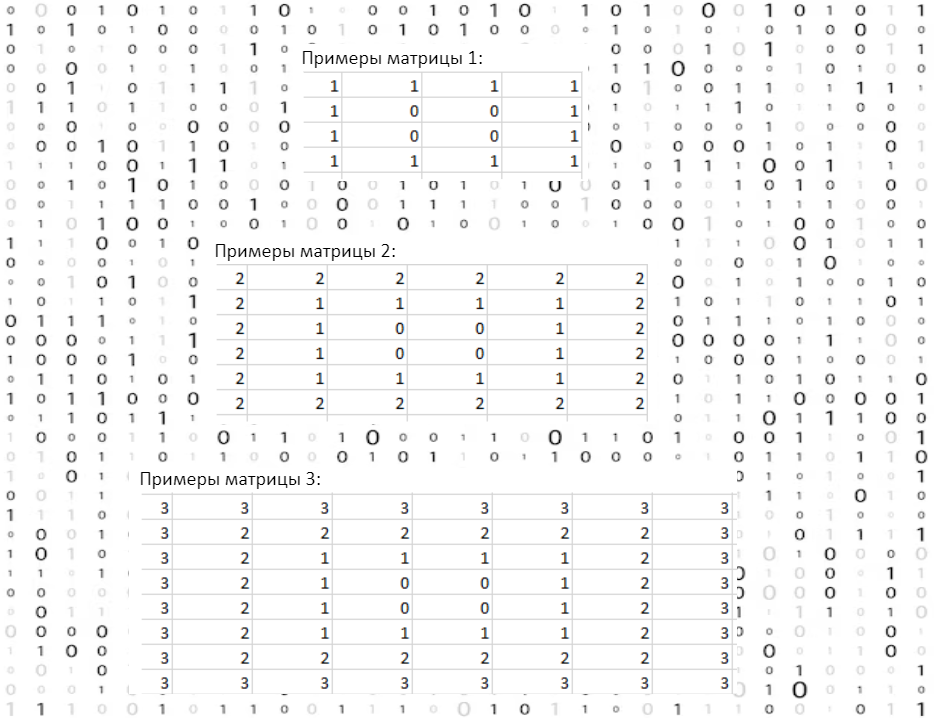
Дан массив NxN. Напишите программу на Java которая находит минимальный элемент диагонали на картинке, без учёта элемента пересечения диагоналей.

**Решение:**

class Solution3 {  
 public int minElementTofDiagonal(int[][] array) {  
 int result = Integer.*MAX\_VALUE*;  
 int size = array.length - 1;  
 int temp;  
 int restriction = array.length / 2;  
 if (array.length % 2 == 0) {  
 for (int i = 0; i < array.length; i++) {  
 temp = array[size - i][i];  
 if (temp < result) result = temp;  
 }  
 } else {  
 for (int i = 0; i < array.length; i++) {  
 temp = array[size - i][i];  
 if (temp < result  
 && i != restriction) {  
 result = temp;  
 }  
 }  
 }  
 return result;  
 }  
  
}  
  
public class Task3 {  
 public static void main(String[] args) {  
 Solution3 solution3 = new Solution3();  
 int[][] array1 = {{1, 2, 3, 4, 5},  
 {5, 7, 9, 2, 1},  
 {0, 9, 1, 8, 7},  
 {6, 3, 6, 6, 6},  
 {99, 100, -2, 3, 1}};  
 System.*out*.println(solution3.minElementTofDiagonal(array1));  
 int[][] array2 = {{1, 2, 3, 4, 5, 1},  
 {5, 7, 9, 2, 1, 0},  
 {0, 9, 1, 8, 7, 5},  
 {6, 3, 6, 6, 6, -5},  
 {99, 100, -2, 3, 1, -4},  
 {99, 100, -2, 3, 1, 8}};  
 System.*out*.println(solution3.minElementTofDiagonal(array2));  
 }  
}

**Задание 4.**

Создать алгоритм заполнения матрицы согласно схемам, приведенным ниже:

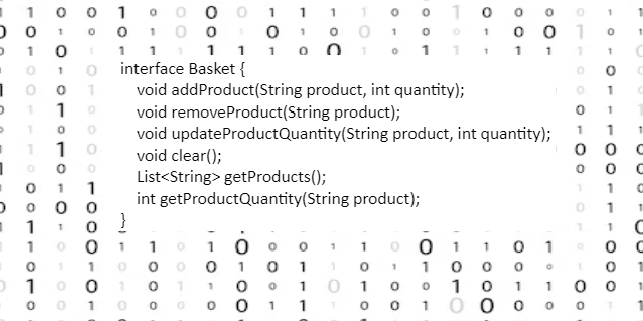


**Решение:**

class Solution4 {  
 int[][] result;  
  
 public int[][] FillArray(int size) {  
 result = new int[size][size];  
 if (size % 2 != 0) {  
 throw new RuntimeException("Не допустимый размер матрицы. " +  
 "Введите четное число");  
 }  
 int top = 0,  
 bottom = size - 1,  
 left = 0,  
 right = size - 1;  
 while (true) {  
 if (left > right)  
 break;  
 // fill top row  
 for (int i = left; i <= right; i++)  
 result[top][i] = size / 2 - top - 1;  
 top++;  
 if (top > bottom)  
 break;  
 // fill right column  
 for (int i = top; i <= bottom; i++)  
 result[i][right] = right - size / 2;  
 right--;  
 if (left > right)  
 break;  
 // fill bottom row  
 for (int i = right; i >= left; i--)  
 result[bottom][i] = bottom - size / 2;  
 bottom--;  
 if (top > bottom)  
 break;  
 // fill left column  
 for (int i = bottom; i >= top; i--)  
 result[i][left] = size / 2 - left - 1;  
 left++;  
 }  
 return result;  
 }  
  
 public void printArray(int[][] array) {  
 for (int[] row : array) {  
 for (int el : row)  
 System.*out*.print(el + " ");  
 System.*out*.println();  
 }  
 }  
}  
  
public class Task4 {  
 public static void main(String[] args) {  
 Solution4 solution4 = new Solution4();  
 solution4.printArray(solution4.FillArray(10));  
 }  
  
}

**Задание 5.**

Реализовать класс корзины интернет-магазина по следующему интерфейсу:



**Решение:**

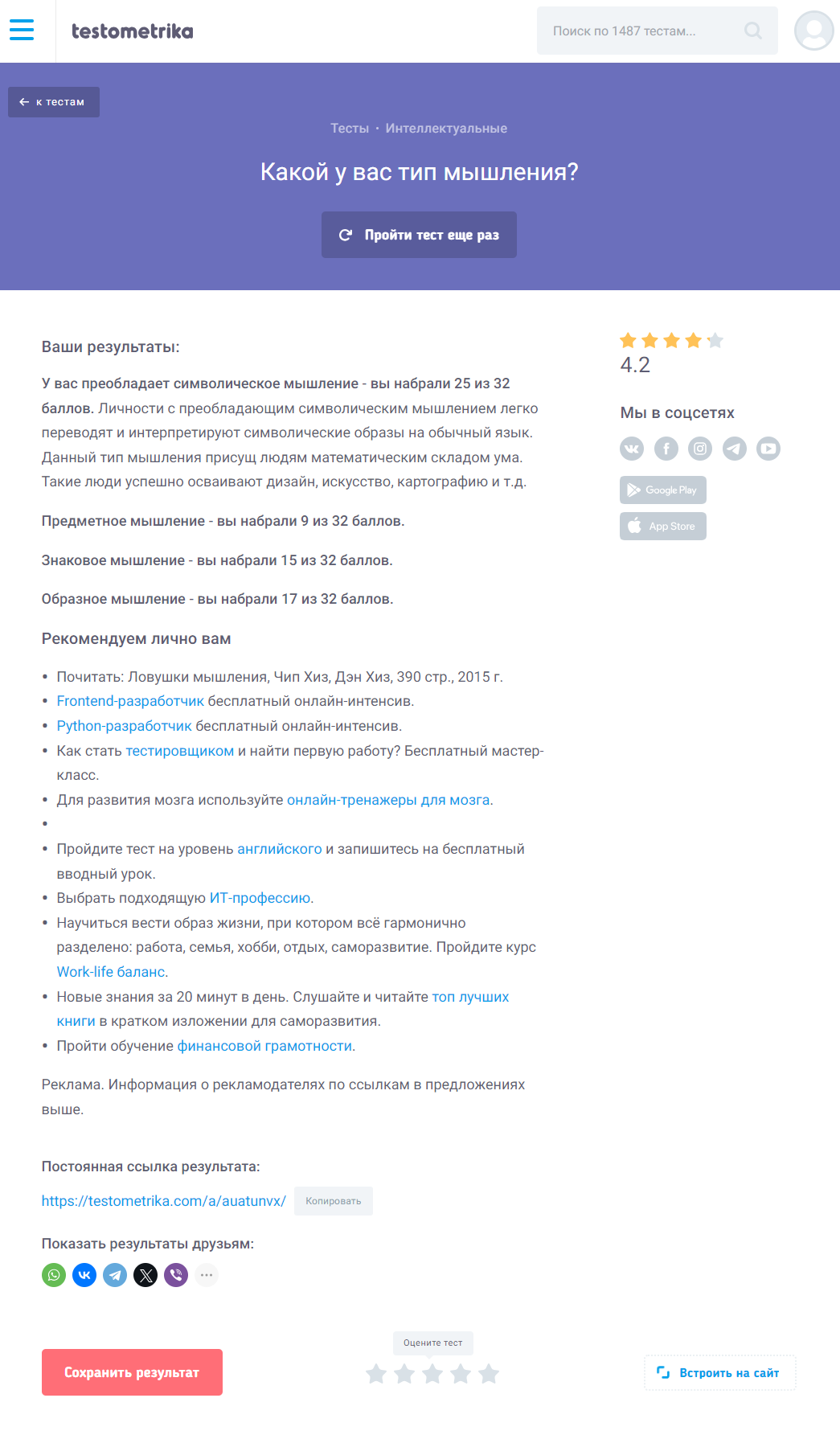
interface Basket {  
 void addProduct(String product, int quantity);  
  
 void removeProduct(String product);  
  
 void updateProductQuantity(String product, int quantity);  
  
 void clear();  
  
 List<String> getProducts();  
  
 int getProductQuantity(String product);  
}  
  
class Product {  
 String title;  
 int quantity;  
  
 public Product(String title, int quantity) {  
 this.title = title;  
 this.quantity = quantity;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
 public int setQuantity(int quantity) {  
 return this.quantity = quantity;  
 }  
  
 @Override  
 public String toString() {  
 return "Продукт{" +  
 "Наименование=" + title +  
 ", Количество=" + quantity + "}\n";  
 }  
}  
  
class BasketImpl implements Basket {  
 List<Product> products = new ArrayList<>();  
  
 @Override  
 public void addProduct(String product, int quantity) {  
 products.add(new Product(product, quantity));  
 }  
  
 @Override  
 public void removeProduct(String product) {  
 products.removeIf(p -> p.title.equals(product));  
 }  
  
 @Override  
 public void updateProductQuantity(String product, int quantity) {  
 removeProduct(product);  
 addProduct(product, quantity);  
 }  
  
 @Override  
 public void clear() {  
 products.clear();  
 }  
  
 @Override  
 public List<String> getProducts() {  
 return products.stream().map(Product::getTitle).toList();  
 }  
  
 @Override  
 public int getProductQuantity(String product) {  
 return getProductByTitle(product).getQuantity();  
 }  
  
 public Product getProductByTitle(String product) {  
 return products  
 .stream()  
 .filter(p -> p.title.equals(product))  
 .findFirst()  
 .orElseThrow();  
 }  
  
 @Override  
 public String toString() {  
 return "Корзина{" +  
 "Продукты=" + products +  
 '}';  
 }  
}  
  
public class Task5 {  
 public static void main(String[] args) {  
 BasketImpl basket = new BasketImpl();  
 basket.addProduct("Молоко", 10);  
 basket.addProduct("Сок", 14);  
 basket.addProduct("Йогурт", 8);  
 System.*out*.println(basket);  
 System.*out*.println(basket.getProducts());  
 System.*out*.println(basket.getProductQuantity("Сок"));  
 basket.updateProductQuantity("Сок", 4);  
 System.*out*.println(basket);  
 basket.removeProduct("Сок");  
 System.*out*.println(basket);  
 basket.addProduct("Сыр", 1);  
 System.*out*.println(basket);  
 basket.clear();  
 System.*out*.println(basket);  
 }  
}

**Задание 6.**

Пройти тест по ссылке и вставить скриншот экрана с результатами.

<https://testometrika.com/intellectual/what-is-your-mindset/>

**Скриншот:**

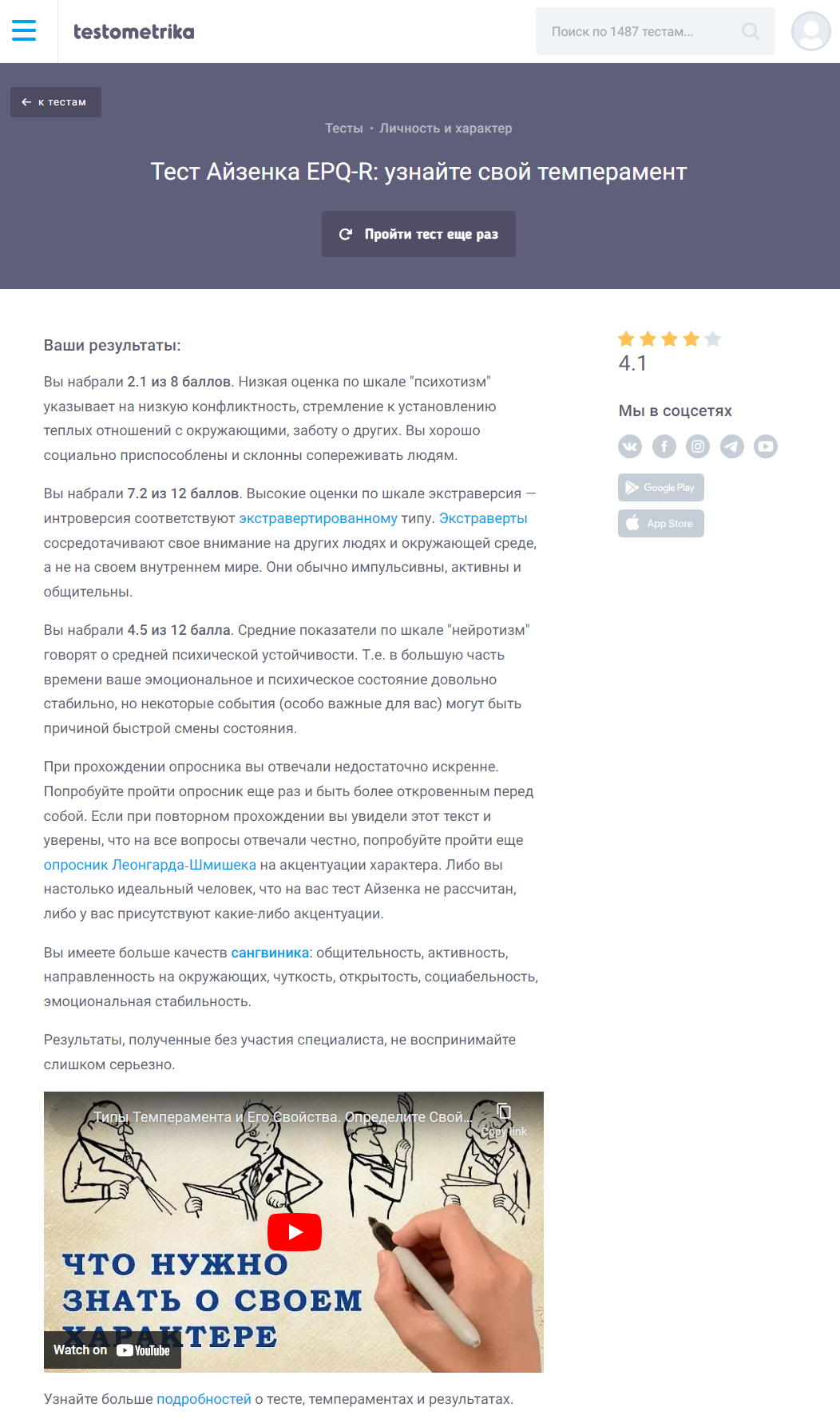


**Задание 7.**

Пройти тест по ссылке и вставить скриншот экрана с результатами.

<https://testometrika.com/personality-and-temper/questionnaire-eysenck-pen/>

**Скриншот:**

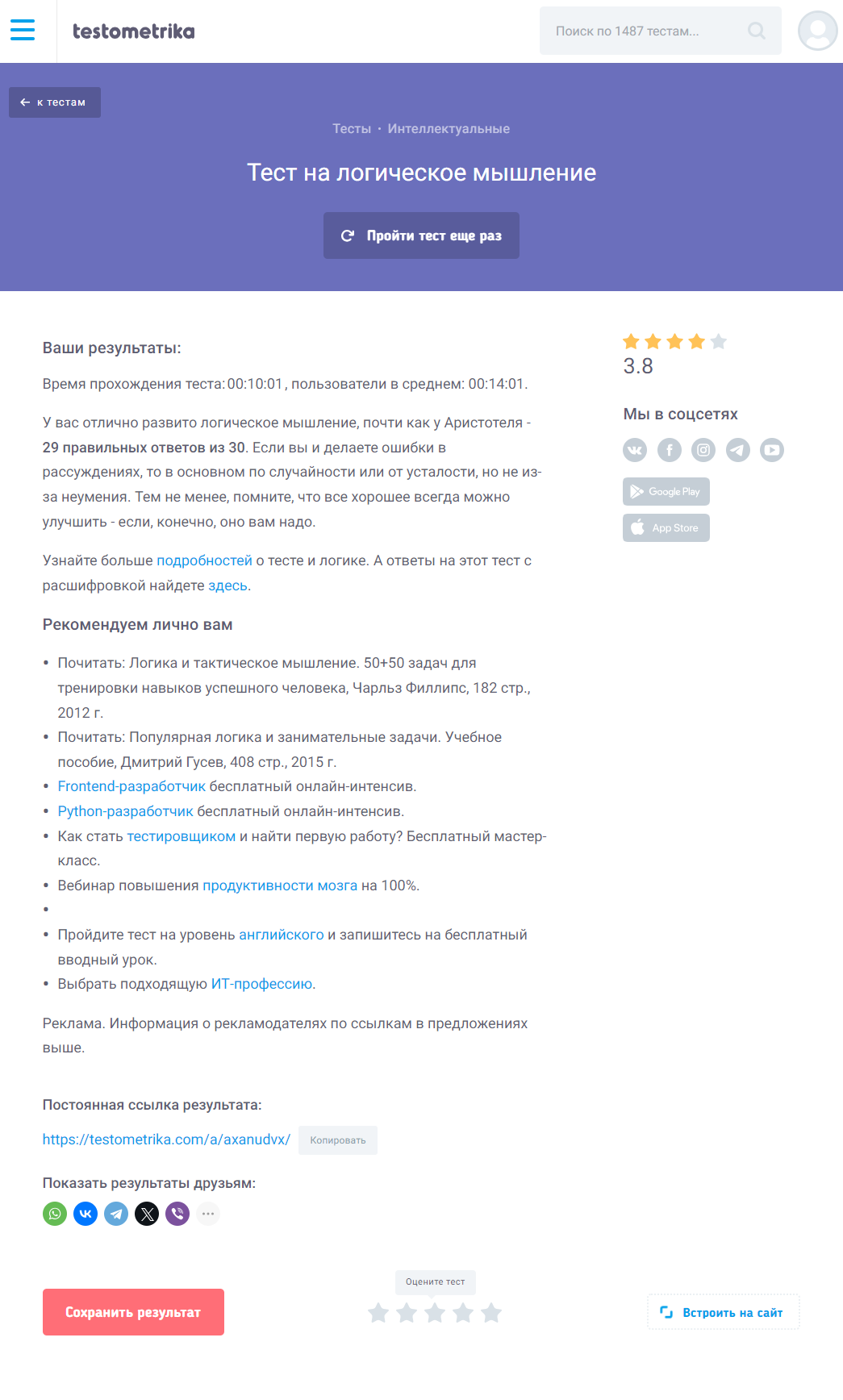
****

**Задание 8.**

Пройти тест по ссылке и вставить скриншот экрана с результатами.

<https://testometrika.com/intellectual/test-of-logical-thinking/>

**Скриншот:**



**ВНИМАНИЕ!**

В случае если вы прошли отбор, наши коллеги с Вами свяжутся и назначат собеседование.  
На собеседовании будет производится оценка ваших технических навыков и софт-скиллов.

**С целью проверки вашей способности учиться, самостоятельно находить и усваивать новую информацию, вам необходимо самостоятельно изучить следующие темы:**

* Collection framework java, иерархия коллекций
* Внутренние механизмы работы коллекций. Понимать как происходит добавления\удаления\хранение элементов. ПОДРОБНО изнутри.
* StreamAPI, типы операций, готовность решить задачу в онлайне.

**ВНИМАНИЕ!**

Наличие веб камеры при прохождении собеседования, а также возможность расшарить экран с открытой средой разработки intellij idea, являются обязательными условиями. В случае невыполнения данных требований мы не сможем принять Вас на курсы.

Спасибо за уделенное время!