WELCOME TO UDT TEST, TRY ALL OF WHAT YOU CAN DO.

SECTION I: Algorithm. (10 Points for basic case, 20 Points for all cases)

1. A logistic company plan to rent a large amount of empty container.

+ Your task is to design an algorithm to help logistic company able to rent **enough containers** (highest priority) at the lowest price.

+ Case 1:

- Input:

```
const neededContainer = 3;
   const listings = [
      {
        name: "Container renter A",
        container: 1,
        totalCost: 1,
      },
      {
        name: "Container renter B",
        container: 2,
        totalCost: 1,
      },
      {
        name: "Container renter C",
        container: 3,
        totalCost: 3,
      },
    ];
```

- Output:

```
[Contract with] Container renter B 2 container, price: 1
[Contract with] Container renter A 1 container, price: 1
[Summary] total cost 2
```

- Explain: The optimal price is to rent 1 container from renter A and 2 containers from renter B, the total cost of them is 2. (Same total cost but the different provider is accepted)

+ Case 2:

- Input:

```
const neededContainer = 10;
const listings = [
  {
    name: "Container renter A",
    container: 5,
    totalCost: 5,
  },
  {
    name: "Container renter B",
    container: 2,
    totalCost: 10,
  },
  {
    name: "Container renter C",
    container: 2,
```

```
            totalCost: 3,
          },
        ];
```

- Output:

```
        [Contract with] Container renter A 5 container, price: 5
        [Contract with] Container renter C 2 container, price: 3
        [Contract with] Container renter B 2 container, price: 10
        Not enough containers
        [Summary] total cost 18
```

- Explain: Display "not enough containers" if don't have enough container providers.

+ Case 3:

- Input:

```
        const neededContainer = 10;
        const listings = [
          {
            name: "Container renter A",
            container: 5,
            totalCost: 5,
          },
          {
            name: "Container renter B",
            container: 2,
            totalCost: 10,
          },
          {
            name: "Container renter C",
            container: 10,
            totalCost: 3,
          },
        ];
```

- Output:

```
        [Contract with] Container renter C 10 container, price: 3
        [Summary] total cost 3
```

CASE STUDY: We are planning to build a backend for an e-commercial platform, your task is doing from system design, implementation to deploy production.

SECTION I:

1. Design ERD for the backend of these features based on best practices to ensure scalable, easy coding. **(3 points)**

+ Customer:
    - Can storage personal information (name, address, email, phone number, gender, etc...), cart, transaction, billing.
+ Agency:
    - Can storage personal information (name, address, email, phone number, gender, etc...), product, transaction, billing.
+ Admin:
    - Can read/create/update/delete agency.

SECTION II:

1. With ERD you have already designed on question 1, what database are you using to implement? **(1 Point)**
2. Why are you using that? What is the strong and weak point of it? **(1 Point)**
3. Write docker-compose.yml to start the database locally. **(1 Point)**
4. Setup Spring **(1 Point)**
5. Using UML on question 1, set up API for these features. **(5 Points)**

SECTION III:

1. Write a sequence diagram to build a solution for authentication and authorize adapt the list of features below. **(5 Points)**

+ Customer:
  - Can log in, log out.
  - Read transaction, billing linked with product.
  - Read agency information.
+ Agency:
  - Agency can read/create/update/delete of own product.
  - Agency can read its own transactions, own billing.
+ Admin:
  - Read billing, transaction, product, customer, agency.
+ Common:
  - User can store auth state after reopening the browser.

2. Using the solution on question 1 implements these features. **(15 Points)**

3. What are the strong and weak points of your solution? How to improve that? **(2 Points)**

4. Build a solution for testing, ensure correct permission scalable from 100 APIs to 1000 APIs. **(5 Points)**

SECTION IV: Good job, right now our application needs to synchronize products, pricing of the Agency by using third-party API.

1. Write a sequence diagram to build a solution to save, merge products data from third-party API to our database. (Third-party API data change every hour) **(8 Points)**

2. What are the strong and weak points of your solution? How to improve that? **(2 Points)**

SECTION V:

1. Write an architecture diagram to build a solution adapt the list of features below.

+ Ensure isolating development and production data. (Don't merge data together) **(3 Points)**
+ Apply Gitlab CI or Github Action to test, build and deploy to VPS automatically. **(5 Points)**