Dumbo Challenge Partner instructions

Useful links

- Hot Line
- Sample HTTP servers to download

Round 1

Technical instructions

- To be able to compete, you will need to start an http server in your local machine. Some sample servers are already available in an archive on top of this document. You **don't have to install any http server** like Tomcat, Apache, Nginx, the sample servers are themselves HTTP servers.
- Kwik-E-Mart will start a central server which will send requests to each challenger's server. As Kwik-E-Mart communicates the URL for the server dashboard, go there and register your local server with your local IP address and the port under your http server is listening on (URL example: http://your_IP_address:port_of_your_http_server)
- The central server will start sending orders to your local server like this:

```
POST /order HTTP/1.1
{
    "prices": [65.6,27.26,32.68],
    "quantities": [6,8,10],
    "country": "IE",
    "names": ["Wine","Apple","Shoes"],
    "reduction":"STANDARD"
}
```

The central server will send this request with `Content-Type: application/json` header.

- Your job is to calculate the amount of the received orders and answer with a JSON object bill, i.e.: { "total": 1000.0 } (**the server checks responses using two decimal digits of precision**, so, i. e., 10.1234 and 10.12 are equal). The central server will expect that you also respond with `Content-Type: application/json` header.
- Your score will be shown in the dashboard.
- The server will send you feedback based on what you have responded. So check if your local http server already handles POST /feedback and, *if not, implement it, otherwise you will not be able to figure out what is going on with your responses*. Here is an example of a feedback the central server can send to you:

```
POST /feedback HTTP/1.1
{
    "type": "ERROR",
    "content": "The field \"total\" in the response is missing."
}
```

Game rules

If your answer match the expected bill, then congratulations: you earned the amount of the bill!
If you answer a total that does not match the expected bill, you will be charged with 50% of the amount of the right bill.
Your answer will be ignored **with no penalty**, if one of these conditions are met:

- your server is unreachable or offline
- you responded with an HTTP code different than 200
- your answer does not match the expected JSON structure: no "total" attribute

Functional description

To calculate the bill, you need to consider the tax of the country from which the order came from and the reduction.

Taxes

Here is the current tax table used in the challenge:

Country	***Code***	***Tax***
Germany	DE	20%
United Kingdom	UK	21%
France	FR	20%
Italy	IT	25%
Spain	ES	19%
Poland	PL	21%
Romania	RO	20%
Netherlands	NL	20%
Belgium	BE	24%
Greece	EL	20%
Czech Republic	CZ	19%

Portugal	PT	23%
Hungary	HU	27%
Sweden	SE	23%
Austria	AT	22%
Bulgaria	BG	21%
Denmark	DK	21%
Finland	FI	17%
Slovakia	SK	18%
Ireland	IE	21%
Croatia	HR	23%
Lithuania	LT	23%
Slovenia	SI	24%
Latvia	LV	20%
Estonia	EE	22%
Cyprus	CY	21%
Luxembourg	LU	25%
Malta	MT	20%

For the order `{"prices":[15.99],"quantities":[1],"country":"ES","reduction":"STANDARD"}`, for example, the response should be `{"total":19.03}`.

Reductions

Following the STANDARD reductions applied for the most part of the orders:

Total	***Reduction***
>= 50 000 EUR	15 %
>= 10 000 EUR	10 %

>= 7 000 EUR	7 %
>= 5 000 EUR	5 %
>= 1 000 EUR	3 %

For the order `{"prices":[4.1,8.03,86.83,65.62,44.82],"quantities":[10,3,5,4,5],"names":['Wine bottle','Cheese','Cookie','Muffin','Wooden stick'],"country":"AT","reduction":"STANDARD"}`, for example, the response should be `{"total":1166.62}`.

Note that reductions are applied **after** the taxes

Have fun… and deliver :)

Round 2

Technical instructions

You will need to return this JSON object:

```
{
      total: 1000,
      voucher: 10,
      licenses: ['OLDER_THAN_21','HUNTING']
}
```

Game rules

- No cash is earned if
  - you return the wrong voucher
  - or: you return the wrong licenses
  - or: voucher attribute is missing in your answer
  - or: licenses attribute is missing in your answer
- If you return the right voucher and all required licenses, then the team earns the amount of the order, just like before.

Voucher feature description

Some categories of articles grant a voucher. You will have to return the right voucher amount.
For example, if there is ‘Wine’ in your order, then you have to give a voucher of 10 because Wine is a drink, and drinks gives a voucher of 10.
Most categories does not need a voucher, so in that case you have to return a voucher

of 0.
If several articles gives a voucher, then you will have to give the best one (only one voucher by order). For example if you have Wine and Cheese, Wine gives 10 and Cheese gives 20, then you have to give 20.
To implement this feature, you will need to call an API that exposes articles categories, and an API that exposes vouchers by category. This data is exposed… somewhere.

Licenses feature description

Some categories of articles require a license in some countries. You will have to warn the customers with the licenses they need.
For example, if there is 'Wine' and 'Gun' in your order, then you have to return OLDER_THAN_21 and HUNTING if the order's country is France, but nothing if the order's country is Syldavia.
If no article require a license, then return an empty array of licenses: `[]`
The order the licenses are presented is no relevant.
To implement this feature, you will need to call an API that exposes articles categories, and an API that exposes licenses required by category in a given country. This data is exposed… somewhere

Example of vouchers & licenses implementation

Given categories repository:

Article	**Category**
Wine	Alcohol
Cookie	Food
Muffin	Food
Wooden stick	Weapons

And vouchers repository:

Category	**Voucher**
Alcohol	10
Cars	20000

And licenses repository:

Category	Country	License required
Alcohol	FR	OLDER_THAN_21
Weapons	FR	HUNTING
Alcohol	XE	OLDER_THAN_21

And order received is:
```
{
     "prices":    [25,25,25,25],
     "quantities":[10,10,10,10],
     "names":['Wine','Cookie','Muffin','Wooden stick'],
     "country":"FR",
     "reduction":"STANDARD"
}
```

When you respond with:
```
{
     total: 1200,
     voucher: 10,
     licenses: ['OLDER_THAN_21','HUNTING']
}
```
Then you earn 1200

Boxes feature description

Rules

You have to return the goods packed into boxes. The goal is to pack the articles using the minimum number of boxes, because each box will cost you money.

There is only 3 possible sizes of boxes : 1V, 3V and 5V.
Every article has a volume expressed in V.
Example: a maintenance bot is 3 in volume, it perfectly fits in a 3V box

If there is a boolean "`vip: true`" in the order, then you will have the choice to:
- return the right boxes filled correctly, and to earn the order total
- or ignore the order: then you won't earn the order total

If there is a boolean “`vip: false`” in the order, then you will not have to return boxes for this order.

If you give an acceptable packaging, then **you earn the total, even if it is not the best possible packaging**, but delivery cost is **deduced** from what you earn. So you have to choose a packaging as cheap as possible.

You earn nothing & a penalty of 50% of the total is applied if :

- an article is missing
- or: there is an article that was not in the order
- or: a box contains is overfilled: articles don’t fit in it
- or: a box contains is underfilled: there is empty space in it
- or: you choose a box size that is not 1, 3 or 5

Your server will get a feedback on the delivery cost, or the penalty.

Box cost model

The cost of boxes depends on the order total.

Box size	**Cost**
1	orderTotal x 0.010
3	orderTotal x 0.012
5	orderTotal x 0.015

Examples of valid packagings you could return

Given the total of the order is $1000,
And the seller return the right total: $1000
And the cost of a box of size 3 is $12
And articles in the order are:

```
articles:   ['Wine bottle','Cheese','Cookie'],
quantities: [            1,       2,       1 ],
volumes:    [            3,       1,       1 ]
```

When the seller returns the articles in these boxes:

```
boxes: [
    {
        size: 3,
        articles: ['Wine bottle']
    },
    {
        size: 3,
        articles: ['Cheese', 'Cheese', 'Cookie']
    }
]
```
Then the seller earns $760

The seller earns $1000 - $12x2 = $976 becauses it uses 2 boxes at $12 each - remember box cost model described before?

The following answer is also valid, but less optimal:
Given cost of size 3 box is $12 and size 1 box is $10
When the seller return these boxes:
```
boxes: [
    {
        size: 3,
        articles: ['Wine bottle']
    },
    {
        size: 1,
        articles: ['Cheese']
    },
    {
        size: 1,
        articles: ['Cheese']
    },
    {
        size: 1,
        articles: ['Cookie']
    },
]
```

Then the seller earns $958

The seller earns less because she uses more boxes. She earns: $1000 - $12 - $10x3 = $958 instead of $976

Over-capacity

Given order total is $1000

When the seller returned the articles in these boxes:

```
boxes: [
     {
          size: 1,
          articles: ['Cheese','Cookie']
     },
]
```

Then the seller loses $500

A penalty of 50% of the order is applied, because the selled tried to put volume of 2 inside a box of size 1. So instead of earning money, the seller loses $500 = 50% of the total.

Under-capacity

Given order total is $1000
And articles volumes are:

```
articles:   ['Wine bottle','Cheese'],
volumes:    [            3,        1 ]
```

When the seller returned the articles in these boxes:

```
boxes: [
     {
          size: 5,
          articles: ['Wine bottle','Cheese'] /* volume = 4 /*
     }
]
```

Then the seller loses $500

A penalty of 50% of the order is applied, because the selled tried to put volume of 4 inside a box of size 5.

Missing article

Given articles in the order are:

```
articles:   ['Cheese','Cookie'],
volumes:    [       1,       1 ]
```

When seller returns these boxes:

```
boxes: [
     {
          size: 1,
          articles: ['Cheese']
     }
]
```

Then the seller loses $500

A penalty of 50% of the order is applied, because the cookie is missing.

Article that were not in the order

Given articles in the order are:

```
articles:    ['Cheese','Cookie'],
volumes:     [      1,        1 ]
```

When seller returns these boxes:

```
boxes: [
     {
          size: 3,
          articles: ['Cheese','Cookie','Chocolate']
     }
]
```

Then the seller loses $500

A penalty of 50% of the order is applied, because no chocolate was ordered.

Bad box size

Given articles in the order are:

```
articles:    ['Cheese','Cookie'],
volumes:     [      1,        1 ]
```

When seller returns these boxes:

```
boxes: [
     {
          size: 2,
          articles: ['Cheese','Cookie']
     }
]
```

Then the seller loses $500

A penalty of 50% of the order is applied, because there is no box of size 2! They only exist in size 1, 3 and 5.

Example of your JSON response if you implement all features

```
{
     total: 12345,
     voucher: 1000,
     licenses: ['OLDER_THAN_21','HUNTING'],
     boxes: [
          {
               size: 3,
               articles: ['Cheese', 'Cookie', 'Chocolate']
          },
          {
               size: 3,
               articles: ['Wine bottle']
          },
          {
               size: 1,
               articles: ['Laser saber']
          }
     ]
}
```