Problem 4. Tech Factory

Input

Our task is to create data base for a factory. The application must have the functionality to fill the data base, test the items and make reports. The factory produces items

A **basic** item will have:

- **id - string**
- **operating system – string**
- **price – floating point number**

There are **only two types of items** - phones and tablets. The phones are – smart phones and cell phones.

Both types of phones **have** the functionality to make and receive calls and send and receive messages. Tablet **does not have** these functionalities.

Smartphone **has the functionality to browse internet**. Every smartphone has **different applications installed** on it.

Tablets **has only** the functionality to stream movies

You will start receiving input data in format:

```
SmartPhone("{id}", "{operating_system}", {price}, ["{app1}", "{app2}", "{app3}"…])
CellPhone("{id}", "{operating_system}", {price})
Tablet("{id}", "{operating_system}", {price})
```

Is it possible a request for basic phone or item to reach the database:

```
Phone/Item ("{id}", "{operating_system}", {price})
```

- You must print **"Can't instantiate abstract class Phone with abstract methods __init__"** and you must not add it in the DB.

Or

- You must print **"Can't instantiate abstract class Item with abstract methods __init__"** and you must not add it in the DB.

Constrains

- Id must **contain only letters** and minimum **length of 8** chars. If the id is invalid print: **"Invalid id!"** and **do not add** this item to the data base
- If price **is 0 or negative** you must print **"Invalid price!"** and **do not add** this item to the data base
- The smart phone should come with **at least 5 applications** if it doesn't print – **"Invalid number of applications!"** and **do not add** the item to the data base

You must receive new items unit command "end" is given.

The second part is to test or make reports for added items in our DB. You will receive input in format:

If the task is to test you will receive:

{test} {id} {functionality}

- If the item **does not have this functionality** or we **do not have in the data base such id** print:

"Invalid request has been made!"

- Functionalities:

- **make_call()** you must print - **Making call...**
- **receive_call()** you must print – **Receiving a call!**
- **send_message()** you must print – **Sending message...**
- **receive_message()** you must print – **Receiving a message!**
- **browse_internet()** you must print - **Browsing...**
- **stream_movie()** you should print - **Streaming movie...**

If you receive:

{report} {kind}

Where kind could be – **SmartPhone, CellPhone, Tablet** you must print information for all items of this kind in the following format:

Item id: {id}
Operating system: {operating_system}

Price: {price}

If kind is **SmartPhone** you must add a row**:**

Applications: {appl}, {app2} …

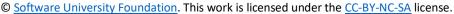*Note: If you receive a non-existent kind or id you must print:* **"Invalid request has been made!"**

Output

When you receive command **"end"** you should end the programm.

Examples

Input	Output	Comments
Tablet("erktmkby", "iOS", 20) CellPhone("abcdefge", "iOS", 20) SmartPhone("abcdefgeee", "iOS", 20, ["facebook", "messanger", "snapchat", "viber", "appStore"]) end test abcdefge stream_movie report Tablet report Android end	Invalid request has been made! Item id: erktmkby Operating system: iOS Price: 20 Invalid request has been made!	Tablet is valid item with valid id and price so we add it to data base. CellPhone is valid item with valid id and price so we add it to data base. SmartPhone is valid item with valid id, price and has 5 or more apps so we add it to data base. The id is valid and the tablet has such function so we print the message from the function.

		The kind (Tablet) is valid so we print informtio)n for all tablets in our Db – in our case just one. ==The kind "Android" is invalid so we print the message for invalid request on the console.==
Input	**Output**	**Comments**
Tablet("abcdege", "iOS", 20) Tablet("abcdegee", "iOS", 0) SmartPhone("abcdefgeeee", "iOS", 20, ["facebook", "messanger", "snapchat", "viber"]) Item("abcdefge", "iOS", 20) end test abcdefge stream_movie end	Invalid id! Invalid price! Invalid number of applications! Can't instantiate abstract class Item with abstract methods __init__ Invalid request has been made!	