☆ Authorization gateway: part 1

Authorization gateway: part 1

If an Affirm customer wants to make a purchase at a store that is not integrated with Affirm, we issue them a virtual credit card. They can input the virtual card number as the payment method when they make the purchase. Affirm then receives authorization requests through the credit card network, asking to approve a purchase of some amount on a card. We must quickly respond with a decision to approve or reject the authorization (auth). You're in charge of creating our system to make auth decisions, to prevent fraud and misuse of cards.

Your system will receive and process two kinds of events: card creation notifications and authorization requests (auths). You will receive a list of these events, each formatted as a valid JSON string. The list is in chronological order, and you will have to maintain the state of the system to process the events correctly. Below is the specification for each event type, and the input/output format for your solution.

Card creation notification

Your system will be called when a new virtual card is created. The components of a card creation notification are as follows:

key	type	meaning
instruction_type	str "card"	this is a card creation notification
card_number	positive integer	the card number
amount_cents	positive integer	the total amount, in cents, which should be authed on this card

Your system will never receive multiple card creation notifications with the same card number. Your system will receive card creation notifications as JSON strings. Here is an example:

```
{"instruction_type": "card", "card_number": 5424181084465199, "amount_cents": 40000}
```

Auths

Your system will also receive auths from the network. The components of an auth are as follows:

key	type	meaning
instruction_type	str "auth"	this is an auth
card_number	positive integer	the card number being authed
amount_cents	nonnegative integer	the amount requested
name	str	the merchant name
industry	str	the merchant industry
country	str	the merchant country

Your system will receive auths as JSON strings. Here are a few examples:

```
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 100000, "name": "Ikea", "industry": "furniture", "country": "USA"}
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 52025, "name": "Uniqlo", "industry": "fashion", "country": "USA"}
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 800, "name": "Coinbase", "industry": "speculative", "country": "Japan"}
```

Auth approval logic

Now that we've gone over the card and the auth events, we're ready to get to the heart of the matter: deciding whether to approve or reject an auth.

When your system receives an auth, it should approve it if

(a) its card number is corresponds to a card that was created in your system

(b) combined with previously approved auths on this card, the total approved amount is no larger than the card amount, and

(c) the fields on the auth do not indicate a high fraud risk. Specifically,

(i) the "country" field must be "USA"

(ii) the "industry" field must not be "speculative" or "precious metals"

Input/output format

Input

For your function, `handle_input_list`, you will receive the input as a list of strings. Each string is valid JSON representing either a card creation notification or auth. The list is in order; that is, instructions occur in the order they appear in the list.

When writing custom test cases, HackerRank needs the length of the list prepended as a separate line to the list itself. Here is an example of an input in the HackerRank format (the "4" will not be part of the argument list of `handle_input_list`):

```
4
{"instruction_type": "card", "card_number": 5424181084465199, "amount_cents": 40000}
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 100000, "name": "Ikea",     "industry": "furniture",   "country": "USA"}
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 800,    "name": "Coinbase", "industry": "speculative", "country": "Japan"}
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 22025,  "name": "Uniqlo",   "industry": "fashion",     "country": "USA"}
```

Output

Output a list of decisions corresponding to the auths in the input list, in the same order as the input auths. Each decision should be formatted as a valid JSON string, which should be the same as the corresponding auth with an additional boolean JSON field, with the key "approved" and the value true if the auth was approved, and false if it was rejected. For example, given the example input above, the correct output is as follows:

```
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 100000, "name": "Ikea",     "industry": "furniture",   "country": "USA", "approved": false}
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 800,    "name": "Coinbase", "industry": "speculative", "country": "Japan", "approved": false}
{"instruction_type": "auth", "card_number": 5424181084465199, "amount_cents": 22025,  "name": "Uniqlo",   "industry": "fashion",     "country": "USA", "approved": true}
```

Commentary:

☆ Authorization gateway: part 2

Authorization gateway: part 2

Your system has been deployed, and it's effective at restricting the amount approved per card, but many fraudulent transactions are not being blocked by the current logic. Your fraud analyst colleagues request the ability to customize the content of the fraud rules and specify the cards to which they should apply. This entails:

1. adding a "category" field to each card.
2. a new rule notification, which specifies a category, and conditions under which cards in that category should be blocked.

Card creation notification (modified)

The components of a card creation notification are as follows (the "category" notification is new):

key	type	meaning
instruction_type	str: "card"	this is a card creation notification
card_number	positive integer	the card number
category	str	the card category (new)
amount_cents	positive integer	the total amount, in cents, which should be authed on this card

Your system will never receive multiple card creation notifications with the same card number. Your system will receive card creation notifications as JSON strings. Here is an example:

```
{"instruction_type": "card", "card_number": 5424181084465199, "category": "clothing_offer", "amount_cents": 40000}
```

Rule creation notifications (new)

Your system will keep track of rules. Rules tell your system to reject certain auths, while approving others, to prevent fraud and misuse of cards. The components of a rule creation notification are as follows:

key	type	meaning
instruction_type	str: "rule"	this is a rule
category	str	category of card to which this rule applies
field	str: {"name", "location", "industry"}	the field on the auth that the rule is checking
values	list[str]	a list of values
rule_type	str: {"block_matches", "block_non_matches"}	the rule type

Your system will receive rules as JSON strings. Here are a couple examples:

```
{"instruction_type": "rule", "category": "default_unlocked", "field": "name", "values": ["Payday Lending, Inc.", "Ponzi Corp"], "rule_type": "block_matches"}
{"instruction_type": "rule", "category": "clothing", "field": "industry", "values": ["fashion", "footwear"], "rule_type": "block_non_matches"}
```

There could be multiple rules created for a given category and/or field.

☆ Authorization gateway: part 3

Authorization gateway: part 3

Your new rule logic has greatly cut down on fraud, while allowing some card categories to be locked to specific use cases. Impressed with your quick execution, your fraud analyst colleagues have a couple more requests:

1. the ability to specify that a category is a subcategory of another
2. the ability to add a blacklist flag to a rule: if a rule with this flag blocks an auth on a card, all later auths should be rejected

Note: we do not expect you to have time to implement both additions; start with whatever feels most doable and work from there.

Card creation notification (unchanged)

This is unchanged from the previous section, part 2 -- see that section for the details.

Rule creation notification (modified)

There is one new component in a rule: an optional blacklist field that will be true if present. If the "blacklist" field is present, the rule is a blacklist rule. (side note: This formulation makes sure you can pass test cases without blacklist rules if you haven't implemented blacklist rules.)

key	type	meaning
instruction_type	str: "rule"	this is a rule
category	str	category of card to which this rule applies
field	str: {"name", "location", "industry"}	the field on the auth that the rule is checking
values	list[str]	a list of values
rule_type	str: {"block_matches", "block_non_matches"}	the rule type
blacklist	bool: true (optional)	whether this is a blacklist rule (new)

Here is an example:

```
{"instruction_type": "rule", "category": "default", "field": "name", "values": ["Check N Go", "Unscrupulous Payday Lending, Inc."], "rule_type": "block_matches", "blacklist": true}
```

As in part 2, there could be multiple rules created for a given category and/or field.

Auths (unchanged)

This is still unchanged; see part 1 or 2 for details.

Subcategory notification (new)

Categories can be made subcategories of other categories using this notification. The expected components of a subcategory notification are as follows: