Programming exercise

Coding：bitmap, Bitmap

发email

coding：email schedule，面试官可能会让同一天的输出按照特定权重排序

第一部分是Coding，基本上是电话面试的重复，但这次需要用到内置库。我只做对了两部分。面试官看起来有点失望，因为我不太熟悉这些库，查了很多东西。

subscription: list of users {name, plan, begin_date, duration}，要求按顺序发email (plan当天发welcome, -15 days发upcoming expiration，expire当天发expire)
第二问：基础上增加一个change in plans, list of changes {name, new_plan, change_date}
第三问是bonus，第二问的基础上再加一个renew，change list中entry为 {name, extension, change_date}

例如：
users = [{name: A, plan: X, begin_date = 0, duration = 30}, {name: B, plan: Y, begin_date = 1, duration = 15}],
输出
0: [Welcome] A, subscribe in plan X
1: [Welcome] B, subscribe in plan Y
1: [Upcoming expiration] B, subscribe in plan Y
15: [Upcoming expiration] A, subscribe in plan X
16: [Expired] B, subcribe in plan Y
30: [Expired] A, subscribe in plan X

第二问例如：
users = [{name: A, plan: X, begin_date = 0, duration = 30}, {name: B, plan: Y, begin_date = 1, duration = 15}]
changes = [{name: A, new_plan: Y, change_date = 5}]
输出
0: [Welcome] A, subscribe in plan X
1: [Welcome] B, subscribe in plan Y
1: [Upcoming expiration] B, subscribe in plan Y
5: [Changed] A, subscribe in plan Y
15: [Upcoming expiration] A, subscribe in plan Y
16: [Expired] B, subcribe in plan Y
30: [Expired] A, subscribe in plan Y

第三问例如：
users = [{name: A, plan: X, begin_date = 0, duration = 30}, {name: B, plan: Y, begin_date = 1, duration = 15}]

changes = [{name: A, new_plan: Y, change_date: 5}, {name: B, extension: 15, change_date: 3}]
输出
0: [Welcome] A, subscribe in plan X
1: [Welcome] B, subscribe in plan Y
1: [Upcoming expiration] B, subscribe in plan Y
3: [Renewed] B, subscribe in plan Y
5: [Changed] A, subscribe in plan Y
15: [Upcoming expiration] A, subscribe in plan Y
16: [Upcoming expiration] B, subcribe in plan Y
30: [Expired] A, subscribe in plan Y
30: Expired] B, subscribe in plan Y

card题

part1
给一串字符代表很多张卡，格式是[len][card][len][card]...
处理字符，根据len提取每张卡，maintain order

part2
在part1的基础上增加一个system suppored的card，return 支持的，maintain order，写到这里开始想起http header那题，就照着类似的思路来了
这部分做完并且run完给的例子，他贴了一堆test case进来，好险，全过，这个时候还是十分钟做part3

part3
在part2的基础上，在字符串里增加generic的卡号，只要system supported里有卡以它开始，就存进结果，这里就是要注意保持order，保持unique

经典汇率题

老汇率转换题
给一个长string，其中包含多种currency的exchange rate。
要求写function，return 某两种currency的汇率。

example input:
USD:CAD:1.26, USD:AUD:0.75, USD:JPY:109.23

example expected output:
CAD:USD => 0.794

我用的双层map去存，key是currency1，value是map of currency2 to rate

Part 1
要求return 某两种currency之间的汇率。这两种currency都在input string中直接相关。
eg，USD:AUD:0.75 and AUD:USD:1.33

Part 2
要求return 某两种currency之间的汇率。这两种currency在input string中不直接相关。
可以通过一个中间的currency相连。
比如说，要求CAD->AUD的汇率。可以通过USD作为中间变量来计算，CAD->USD->AUD.
final rate = 0.75 *（1/1.26）= 0.595

Part 3
最后时间不太够，只能大概捋一下思路。题大概就是说，如果中间要通过多个currency相连。
因为part2是中间只有一个middle currency，part3应该会是不确定有多少个。
solution的话，我觉得可以用graph dfs traverse + backtrack来找。
每进一步，就rate * curRate。如果sub dfs没找到，rate/curRate。
［/hide］

处理csv格式文本

/*
Part 1 prompt: Stripe in Brazil is obliged to register customer's transactions for each merchant with the central bank as an aggregated unit per day.
These are called receivables. A receivable is identified by 3 identifiers:
* merchant_id (String): The id of the merchant on Stripe side.
* card_type (String): The type of the card used for the transaction (e.g. Visa)
* payout_date (String): String date of the funds available to the merchant by Stripe.

A payment transaction in Stripe API can be represented as the following object:
```
Transaction {
   string customer_id
   string merchant_id
   string payout_date
   string card_type
   int amount
}
```

Implement register_receivables function that takes a string in CSV format
where each line represents a transaction and returns the registered aggregated receivables using the rules above.
Print the returned receivables to console using the format below.
Feel free to parse the CSV using a standard or a 3rd party library or implement it yourself.

You can assume the following about the input:
* The first line of the input is a header. The header is always the same so it can be ignored or hardcoded
* You can assume that the input has
already been read from a file and checked for correctness
* No data fields in this file will include commas or whitespace surrounding the field values.

You can also assume the following about the output:
* The first line of the output is the header. The header is always the same so it can be hardcoded
* Order of the output does not matter

Example input 1:
```
customer_id,merchant_id,payout_date,card_type,amount
cust1,merchantA,2021-12-30,Visa,150
cust2,merchantA,2021-12-30,Visa,200
cust3,merchantB,2021-12-31,MasterCard,300
cust4,merchantA,2021-12-30,Visa,50
```

Output 1:
```
merchant_id,card_type,payout_date,amount
merchantA,Visa,2021-12-30,400
merchantB,MasterCard,2021-12-31,300
```

Example input 2:
```
customer_id,merchant_id,payout_date,card_type,amount
cust1,merchantA,2021-12-29,MasterCard,50
cust2,merchantA,2021-12-29,Visa,150
cust3,merchantB,2021-12-31,Visa,300
cust4,merchantB,2021-12-29,MasterCard,200
```

Output 2
```
merchant_id,card_type,payout_date,amount
merchantA,MasterCard,2021-12-29,50
merchantA,Visa,2021-12-29,150
merchantB,Visa,2021-12-31,300
merchantB,MasterCard,2021-12-29,200
```
*/
part2

/*
In Brazil, settlement times can take up to 30 days for domestic card transaction. i.e. A merchant selling items online
will receive their money from a customer after a month of selling an item. This has created a need where merchants
are looking for ways to receive their money faster.

Per the regulations, merchants can sell their receivables to a financial institution. The financial institution
will pay the funds to the merchant earlier and receive the funds from Stripe instead on the payout date.

An agreement between the merchant and a financial institution is called a contract. Stripe is obliged to respect
those contracts and update the registered receivables. Each contract is mapped to one receivable based on the
same 3 identifiers as above:
* merchant_id (String): The id of the merchant on Stripe side.
* card_type (String): The type of the card used for the transaction (e.g. Visa)
* payout_date (String): String date of the funds available to the merchant by Stripe.

A contract sent to Stripe is represented as follows:
```
Contract {
    string contract_id
    string merchant_id
    string payout_date
    string card_type
    integer amount
}
```

Implement update_receivables function that takes the list of registered receivables from part 1 and additional parameter of list of contracts.
The result should be the updated list of receivables.
For each contract, a receivable should be created for the contract id, and the corresponding merchant receivable should be removed.

Example input 1:

Transactions:
```
customer_id,merchant_id,payout_date,card_type,amount
cust1,merchantA,2022-01-05,Visa,300
cust2,merchantA,2022-01-05,Visa,200
cust3,merchantB,2022-01-06,MasterCard,1000
```

Contracts:
```
contract_id,merchant_id,payout_date,card_type,amount
contract1,merchantA,2022-01-05,Visa,500
```

=> update_receivables(registered_receivables, input_contracts)

Output 1:
```
id,card_type,payout_date,amount
contract1,Visa,2022-01-05,500
merchantB,MasterCard,2022-01-06,1000
```

Example input 2:
Transactions:
```
customer_id,merchant_id,payout_date,card_type,amount
cust1,merchantA,2022-01-07,Visa,500
cust2,merchantA,2022-01-07,Visa,250
cust3,merchantB,2022-01-08,MasterCard,1250
cust4,merchantC,2022-01-09,Visa,1500
```

Contracts:
```
contract_id,merchant_id,payout_date,card_type,amount
contract1,merchantA,2022-01-07,Visa,750
contract2,merchantC,2022-01-09,Visa,1500
```

=> update_receivables(registered_receivables, input_contracts)

Output 2:
```
id,card_type,payout_date,amount
contract1,Visa,2022-01-07,750
contract2,Visa,2022-01-09,1500
merchantB,MasterCard,2022-01-08,1250
```
*/

Bracket Exapnsion

part 1: expand single brace: prefix {a,c,d}, suffix -> prefixasuffix, prefixcsuffix, prefixdsuffix
Example 1:
Input = "/2022/{jan,feb,march}/report"
List<String> Output = ["/2022/jan/report"
  "/2022/feb/report"
  "/2022/march/report"]

part 2: expand single brace with invalid input cases
If there are less than 2 tokens enclosed within curly braces or incorrect expression
(eg. opening and closing braces not present, only opening brace present,
closing brace present before opening brace etc) return the output same as input
these patterns do not have enough tokens, so they are not expanded:
Example 1:
Input: pre{mid}suf
Output: pre{mid}suf
Example 2:
Input: pre{}suf
Output: pre{}suf

part 3: 离扣 1087，没时间了，但说了下思路

Debug

Debug：mako, requests

第二部分是找bug。老实说，这个面试官有点让我不爽。首先你得找到bug。当时他不给我30秒思考的时间，如果我安静地停顿了5秒，他就会插话告诉我接下来该做什么。有时我正说出我准备检查的下一个点，他就插嘴说出来，而且还是我刚要说的内容！不知道是不是他的性格，但我希望这没有让我看起来很糟。最后他稍微提示我一下bug在哪，我立刻修复了，然后他又加了个小问题，我也很快解决了。

Debug：给40分钟去调试错误的test case，我在第39分钟完成。是一个Json解析的library，getPath的结果不正确；

Integration

bikemap
第三部分是集成。其实就是写一些POST请求代码，针对已有的API，并且有很多部分。我做完了四部分，然后时间不够。不知道他们是否期待你做完更多，但我对这一轮最有信心。这个面试官没怎么干涉我，这对我来说最好，因为我需要安静才能思考。

Integration：调用一个API，解析结果；第二题是构造特定的payload，然后下载一个图片的map；第三个构造的payload需要先解析一个文件，并且构造的逻辑更加复杂；

SD

第五轮 SD：Ledger service

这轮系统设计，Stripe 玩的是差异化。别家可能重点看宏观架构，但 Stripe 死磕 API 设计细节。一上来别着急画架构图，先和面试官把服务的核心功能、性能要求、使用场景全聊透。比如，这个 Ledger service 是给内部用还是对外提供？吞吐量要到多少？数据一致性怎么保证？
聊清楚需求后，针对性设计 service 层和数据库。API 设计别含糊，参数类型、返回值格式、错误码定义，这些细节都得有。数据库设计也别只说用啥数据库，表结构怎么设计、索引怎么建，都得讲明白。记住，全程和面试官保持沟通，边设计边确认方向，别闷头搞完才发现走偏了。
sd：metric counter

System design：一个library，支持metric的收集。因为我工作中就用到类似的library，所以我回答的很顺手。