Node.js and Docker Crypto/Payments Test

At a crypto payments company receives thousands of deposits from customers per day. This test is designed to test your ability to work with a transaction set that could get returned by a blockchain daemon like bitcoind.

The data we work with in this scenario comes from bitcoind's rpc call `listsinceblock`. A frequently used approach to detect incoming deposits is to periodically call `listsinceblock` and process the returned data. This test contains 2 json files that represent the data from 2 separate calls to this endpoint. Your task is to write code that processes those files and detects all valid incoming deposits.

These instructions do not specify every single detail you should take into consideration. This is done on purpose to test your ability to analyze a problem and come up with a reasonable and safe approach. Keep in mind that your code will determine how much money each customer will get. Thoroughness is one of the most important qualities for this role.

Goal: Process transactions and filter them for valid deposits.

Note: A deposit is considered valid when it has at least 6 confirmations.

Known customer addresses are:

- Wesley Crusher: mvd6qFeVkqH6MNAS2Y2cLifbdaX5XUkbZJ
- Leonard McCoy: mmFFG4jqAtw9MoCC88hw5FNfreQWuEHADp
- Jonathan Archer: mzzg8fvHXydKs8j9D2a8t7KpSXpGgAnk4n
- Jadzia Dax: 2N1SP7r92ZZJvYKG20NtzPwYnzw62up7mTo
- Montgomery Scott: mutrAf4usv3HKNdpLwVD4ow20LArL6Rez8
- James T. Kirk: miTHhiX3iFhVnAEecLjybxvV5g8mKYTtnM
- Spock: mvcyJMiAcSXKAEsQxbW9TYZ369rsMG6rVV

Requirements

Build a dockerized Node.js application to process the two transaction sets.

If you're not comfortable with Node.js, feel free to use the language of your choice.

The command `docker-compose up` **MUST**:

1. Read all transactions from `transactions-1.json` and `transactions-2.json` and store all deposits in a database of your choice.

2. Read deposits from the database that are good to credit to users and print the following 10 lines on stdout:

```
Deposited for Wesley Crusher: count=n sum=x.xxxxxxxx
Deposited for Leonard McCoy: count=n sum=x.xxxxxxxx
Deposited for Jonathan Archer: count=n sum=x.xxxxxxxx
Deposited for Jadzia Dax: count=n sum=x.xxxxxxxx
Deposited for Montgomery Scott: count=n sum=x.xxxxxxxx
Deposited for James T. Kirk: count=n sum=x.xxxxxxxx
Deposited for Spock: count=n sum=x.xxxxxxxx
Deposited without reference: count=n sum=x.xxxxxxxx
Smallest valid deposit: x.xxxxxxxx
Largest valid deposit: x.xxxxxxxx
```

The numbers in lines 1 - 7 **MUST** contain the count of valid deposits and their sum for the respective customer.

The numbers in line 8 **MUST** be the count and the sum of the valid deposits to addresses that are not associated with a known customer.

Note: We'll match for these 10 lines with regular expressions. Please stick to this exact template, otherwise it won't be detected.

Submitting your results

Compress your source code as zip archive and send us a link where we can download it. Sharing via Dropbox or Google Drive has worked well in the past. Make sure the Dockerfile is on the top level.