# Search in transaction statistics of cryptocurrency

## Problem definition

Cryptocurrency is getting hot these year. So John Ceena, an oil tycoon wants to dive into the cryptocurrency market. However, the currencies (幣別), exchanges (交易所), and so many things make John really confused. As a result, he wants to hire you helping him to sort out the large data, and find out the best currency, exchange to put his enormous asset in.

Your job:

a. Put the data into a format easy for search
   (for instance, sort the data with multiple keys)

b. Perform several kinds of search, as explained below.

The database (stored in a data file) to be searched has the following fields:

- date: date in the format "yyyymmdd"
- currency: Name of the cryptocurrency (with string length less than 100).
- exchange: Name of the store where transactions occur
  (with string length less than 100)
- low: Lowest price (with data type of float)
- high: Highest price (with data type of float)
- cap: value of transactions (with data type of long long int)

Note that in the data file, columns (fields) are separated by tabs '\t', and each entry (of a given date) is terminated by newline '\n'.

You program should support 4 types of commands for querying the database. Again, items in the query input are separated by tab '\t' (in order to avoid a item with a space in it), while items in the output are separated by a single space. These 4 types of commands are listed as follows.

1. **Simple query: Find the data corresponding to the inputs of [date], [currency], and [exchange].**

```
input: query [date] [currency] [exchange]
output: [low] [high] [cap]
```

[low] & [high] should be float, accurate to the 4th decimal places. Note that you need to use "float" in order to conform to our standard program, which might lose some precision though.

2. **Daily min/max: Find the minimum or maximum on a given date of a given currency.**

   ```
   input:  price [min/max] [date] [currency]
   output: [min/max of the currency on that date]
   ```

   The output should be float, accurate to the 4th decimal places. Note that you need to use "float" in order to conform to our standard program, which might lose some precision though.

3. **Total cap of all currencies, for a given date and a given exchange**

   ```
   input: cap   [date] [exchange]
   output: [total_cap]
   ```

   The output should be long long int.

4. **End of query**

   ```
   input: end
   action: exit the program
   ```

You can safely assume that the command format will always be correct. If your query retrieves nothing, you should simply output "none".

## How to test your program

The data file should be passed as argv[1]. The query file should be passed as standard input. So TA will run your program on a workstation, like this:

```
./myProgram dataFile < inputQueryFile > outputFile
```

## ● Open data for you to test

The open test data files are available for you to test your program:

- They are available at our workstation: "/tmp2/DSA2018Spring/hw2".
- You can also download them from this link, with the following file name convention:
  - *.in: input query files (with 1.in for "query", 2.in for "price", 3.in for "cap", and 4.in for mixed commands)
  - *.out: output files corresponding to input query files
  - *public_data.txt: data file to be searched/queried.

==========================================================

Here are some sample commands based on the given data file:

```
Input: query  20130114  Diamond   Tidex
Output: 2.9440 4.6044 24

Input: price  max 20130114  Diamond
Output: 4.6257

Input: cap   20130114  Tidex
Output: 210671

Input: query  20120114  Roger  Jang
Output: none
```

==========================================================

Note: Except that cap should be stored with "long long int", all the other data should be stored with integer/float.

## ● Requirements & suggestions

- The standard approach to this exercise is to perform multiple-key binary search.
  - ➢ You can perform stable sort from LSK (least significant key) to MSK (most significant key), as explained in our slides/recordings. For "simple query" and "daily min/max", you need to perform stable sort on keys of [date, currency, exchange]. For "currency of the max total cap...", you need to perform stable sort on keys of [date, exchange]. In other words, you may need to keep 2 different orders for different types of queries.

- Once the data is sorted, you can perform multiple-key binary search.

*Note that during the sort, you should minimize data movement.

- You can also try other methods, such as <map> and <unordered_map> in STL of C++, or any other methods, as long as your program can satisfy the time and memory limits.

- Time and memory limits on judge system: 10sec, 512MB. FYI, TA's program (using fgets, std:sort, binary search, etc., with O2 optimization) takes 4~5 seconds on judge.