Tick Engine

Problem Statement

Design a tick engine that reads a 'tick' file and answers queries on it. The tick engine should read command input from stdin and respond to the following commands by writing to stdout.

- tickfile `<n>`
  - The next `<n>` lines in the stdin represents a 'tick' each, which is a timestamp associated with a symbol and a variable number of fields. The format of each line is:
    - `<timestamp> <symbol> <field name 1> <field value 1> [<field name 2> <field value 2> ...]`
  - The names of the symbols and fields are not known in advance, and there is no maximum to the number of fields any tick can contain. The fields can be in any order.
  - Input lines are guaranteed to be in ascending timestamp order. However, duplicate timestamps may be present.
  - Once the engine is finished processing tickfile, it outputs string literal "tickfile completed".
  - This command will be first command in any test file, and there will be exactly one tickfile command in a test file.
- sum `<start time> <end time> <symbol> <field>`
  - For all ticks between `<start time>` and `<end time>` (both inclusive) which have symbol `<symbol>` and field `<field>`, sum the value of `<field>` and print the result.
- product `<start time> <end time> <symbol> <field1> <field2>`
  - Print the sum of `<field1>` * `<field2>` for all ticks between `<start time>` and `<end time>` (both inclusive) that have symbol `<symbol>` and which have entries for both `<field1>` and `<field2>`. Ticks that only have entries for one of the fields should be ignored.
- max `<start time> <end time> <symbol> <field> <k>`
  - For all ticks between `<start time>` and `<end time>` (both inclusive) which have symbol `<symbol>` and field `<field>`, print the biggest `<k>` values separated by single whitespace in the descending order. If number of values in the given time range is less than `<k>`, then print all of them in the descending order.
- delta `<symbol> <field> <k>`
  - Consider the ticks associated with symbol `<symbol>` and field `<field>` as a time series. Partition this time series into a number of segments. Each segment consists of contiguous values from the time series: $S((t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n))$. The aim is to approximate each segment with a linear function $L(y = at + b)$. The cost associated with the approximation for a segment is given below, where k is the penalty of adding a new segment:

    $$\sum_{i=1}^{n} (y_i - at_i - b)^2 + k$$

    Compute the partition of the time series into a number of segments such that summation of cost associated with each segment is minimal. Print the ceiling of the resulting cost (i.e. the smallest following integer).

There are multiple test files which include different set of queries and tickfile data. Each test file contains tick data and has only a single type of queries. For each type of the queries specified above, there will be at least one test file containing only those types of queries. You would be scored **separately on each test file** and also on the **aggregated measure from all the test files**. You would be graded for a test file

only if you answer all the queries in the test file correctly.

Constraints and Assumptions:

- `<start time>` and `<end time>` are 32-bit unsigned integer times.
- `<timestamp>` is an unsigned 32-bit integer timestamp representing seconds since the EPOCH.
- `<symbol>`, `<field name 1>`, `<field name 2>` ... are alphanumeric strings without spaces.
- `<field value 1>`, `<field value 2>` ... are 32-bit unsigned integer values.
- `<n>` and `<k>` are 32-bit positive integer values.
- For a `<start time>` or an `<end time>` in a query, if there are multiple ticks in the tick file with these timestamps, take the first occurrence of `<start time>` and the last occurrence of `<endtime>` as it appears in the tickfile for your calculations.

Please note:

It is highly encouraged for participants to describe the algorithm they have used, the time and space complexities for the same and the assumptions they have taken for each of the queries they have implemented in the form of code comments.

Input Format

Tick File

tickfile n

timestamp1 symbol1 field1 value1 field42 value42

timestamp2 symbol10 field10 value10 field4 value4 field80 value80

.

.

timestampn symbol10 field10 value10

sum timestamp1 timestamp3 symbol1 field5

sum timestamp2 timestamp4 symbol42 field42

Keep two things in mind:

- There can be variable number of fields in a tick and fields can be in any order.
- Timestamps can repeat.

Output Format

Output file

tickfile completed

sum1

sum2

Sample Input

tickfile 5

10 s1 f1 100 f3 120

10 s2 f1 200 f2 210 f3 220

11 s1 f2 110

12 s1 f3 121

15 s2 f2 211 f3 221

max 10 15 s1 f1 2

max 10 12 s1 f3 2

max 10 15 s2 f2 3

max 10 10 s2 f1 1

Sample Output

tickfile completed

100

121 120

211 210

200

Explanation

Query: max 10 15 s1 f1 2

Output: 100

Explanation: In the timerange 10 to 15 (both inclusive), f1 field of s1 symbol only ticked once at 10. The query asks you to output 2 biggest f1 values but there exists only one. Hence, it should be your only output.

max 10 12 s1 f3 2

Output: 121 120

Explanation: In the timerange 10 to 15 (both inclusive), f3 field of s1 symbol ticked twice at 10 and 12. The query asks you to output 2 biggest f3 value. Hence, your output should be 121 and 120 in the descending order.

Please keep in mind that

For max query, if the number of maximum values asked is more than the number of fields ticked in the given time range, you should return the field values ticked in the right order. If the symbol and field does not tick in the given time range, then you print an empty line.