3. DVD Elevator Algorithm

Program Name: DVDelevator.java Input File: dvdelevator.dat

You have taken an internship at Flaming Monkey Games working on a launch title for the new Phony PlayBox video game console. The game is currently loading too slowly and the lead programmer has asked you to implement an elevator loading algorithm to speed it up.

Data on DVDs are stored in a linear fashion from the inside to the outside of the disk. The laser can move either way across the disk, but always reads data in multiples of sector sizes as it moves from inside to outside. It takes time to move the laser, and the size of the sectors and the amount read per second depends on the speed of the drive. You don't have direct control over the laser; you can only send file read requests to the operating system. You can also assume the operating system starts by having the laser at the beginning of the first sector.

The PlayBox has an optical disk with the following specifications:

- disk size:
  - there are 131,072 sectors with a sector size of 64 kilobytes
  - total disk size is 8 gigabytes of data, where
    - 1024 bytes = 1 kilobyte
    - 1024 kilobytes = 1,048,576 bytes = 1 megabyte
    - 1024 megabytes = 1,073,741,824 bytes = 1 gigabyte
- disk speed:
  - the laser takes 1 microsecond to swipe over a sector in either direction, whether it is reading data or not.
  - overall drive speed is 20 megabytes per second, where
    - 1000 microseconds = 1 millisecond
    - 1000 milliseconds = 1 second

An elevator algorithm is used to order requests based on the motion of the laser in order to minimize laser motion and increase disk read throughput. Your boss has given you explicit instructions on how to implement the algorithm. The instructions are:

1. By default the algorithm should be in an "idle" state while waiting for read requests.
2. When a request is received, it should wait 10 milliseconds and lump all other requests it receives during that time into a single request batch. Requests received at the exact end time of the batch time window should be included inside the batch.
3. After the time expires it should dispatch all the requests to the operating system in sector order so all the requested data can be read as fast as possible without additional seeks. It should also keep track of the sector the laser will stop on as an optimization for new reads.
4. The system should go back to "idle" until more reads come in.
5. When a new read comes in it should be checked against the current laser sector. If the new read is at or past the current laser sector the request should be dispatched right away. If not then the algorithm should go back to #2.

The lead has asked that you test your code by writing a program that will emulate what will happen in the game by reading in read requests from a file and running them through your algorithm. The program will also have to emulate what a DVD drive will do in order to be accurate. Optimal output will also be provided so that you can make sure your algorithm is as efficient as possible.

Input

The input file will contain an unknown number of lines. Each line will contain 3 numeric values separated by a single space:

- the time in microseconds since the start of the game until the request was made, in ascending order
- the start sector for the request, and
- the end sector for the request.

3. DVD Elevator Algorithm (cont.)

Output

You should output all the requests from the input file, one per line, in the order they are processed by the DVD hardware. Each line should be of the form "`Time X: Y to Z`" where `X` is the time in microseconds since the start of the game until the DVD hardware processes the request, and `Y` and `Z` are the start and end sectors respectively.

Example Input File

```
10 2 200
100 300 400
900 210 220
1100 240 270
10000 220 240
10002 10000 20000
11003 30000 50000
20000 80000 90000
20002 20000 30000
```

Example Output to Screen

```
Time 10011: 2 to 200
Time 10219: 210 to 220
Time 10229: 220 to 240
Time 10249: 240 to 270
Time 10309: 300 to 400
Time 20009: 10000 to 20000
Time 40009: 30000 to 50000
Time 90009: 80000 to 90000
Time 180009: 20000 to 30000
```