**Problem Statement:**

Consider a monitoring system, which monitors 1000 servers. Each server has 2 CPUs. Each server generates a log for CPU usage every minute.

The format is like this:

timestamp IP cpu\_id usage

1414689783 192.168.1.10 0 87

1414689783 192.168.1.10 1 90

1414689783 192.168.1.11 1 93

1. Please write a simulator to generate the logs for one day, say 2014-10-31, just use random numbers between 0% to 100% as CPU usage. The generator should write data to files in a directory.  The timestamp is Unix time.
2. Please create a command line tool which takes a directory of data files as a parameter and lets you query CPU usage for a specific CPU in a given time period. It is an interactive command line tool which reads a user’s commands from stdin.
3. The tool may take several minutes to initialize, but the query result should be returned within 1 second.
4. The tool should support two commands:

* One command will print results to stdout. Its syntax is QUERY IP cpu\_id time\_start time\_end. Time\_start and time\_end should be specified in the format YYYY-MM-DD HH:MM where YYYY is a four digit year, MM is a two digit month (i.e., 01 to 12), DD is the day of the month (i.e., 01 to 31), HH is the hour of the day, and MM is the minute of an hour.
* The second command to support is EXIT.  It will exit the tool.

E.g.

To run the generator:

./generate.sh DATA\_PATH

To run the interactive query tool:

./query.sh DATA\_PATH

>QUERY 192.168.1.10 1 2014-10-31 00:00 2014-10-31 00:05

CPU1 usage on [192.168.1.10](http://192.168.1.10/):

(2014-10-31 00:00, 90%), (2014-10-31 00:01, 89%), (2014-10-31 00:02, 87%), (2014-10-31 00:03,  94%) (2014-10-31 00:04, 88%)

>QUERY 192.168.1.12 0 2014-10-31 00:00 2014-10-31 00:05

CPU0 usage on [192.168.1.12](http://192.168.1.12/):

(2014-10-31 00:00, 90%), (2014-10-31 00:01, 89%), (2014-10-31 00:02, 87%), (2014-10-31 00:03,  94%), (2014-10-31 00:04, 88%)

>EXIT

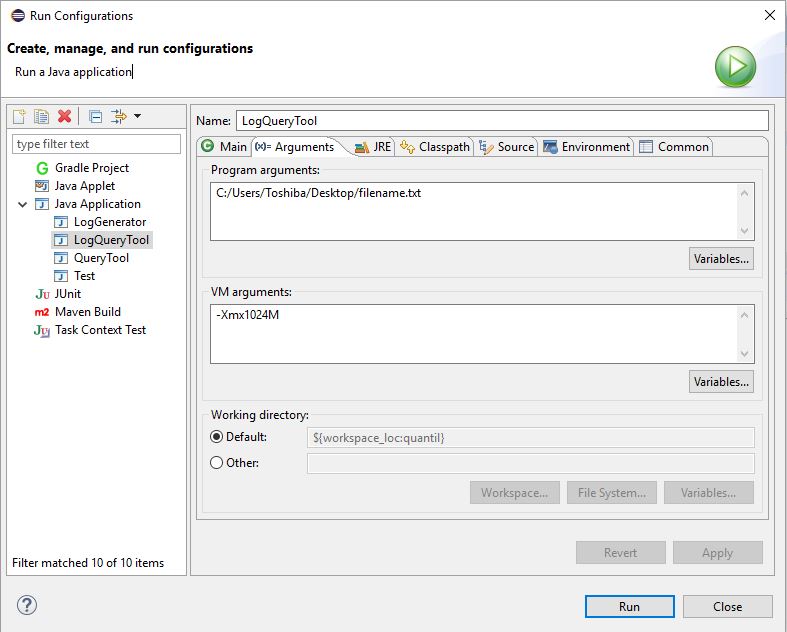
**Solution:**

**Running the code in Eclipse Neon:**

1. Opening java files in eclipse:
   * Create a new java project in eclipse.
   * Right click on the 'src' folder of the pakcage created.
   * Click on 'Import', select 'general', select 'file system'
   * Browse the folder where java files are stored.
   * Select the java files to be imported
   * Click on finish.
   * Delete packages defined in the code if there are any. Let the code run in default package of project created.
2. Executing the code:
   * Set up command line arguments for both the java applications "LogGenerator" and "LogQueryTool"to filepath where log is to be written/ read from.
     1. Click on **Run** -> **Run Configurations**
     2. Click on **Arguments** tab
     3. In **Program Arguments** section, Enter your arguments. (Eg: C:/Users/Toshiba/Desktop/filename.txt)
     4. Click **Apply**
   * Increase Heap Memory allowed for the code to run. (How to?- Refer next section - subsection 5)
   * First execute "LogGenerator" which writes log data into the text file entered.
   * Then run "LogQueryTool" which is the main tool to search through the log file.

**Assumptions:**

1. The log generator is programmed to generate log only for one day, i.e **2016-July-09**.
2. For 1000 servers the valid IP range is **198.162.0.0 to 198.162.3.231. i.e 1000 IPs.**
3. The time entered should be in terms of a **24 hours clock and not 12 hours**.
   1. That is, for 1 PM, please enter 13:00.
   2. For 12 AM, please enter 00:00. And so on.
4. For simulation purpose, to generate log every minute of the day, in class **LogGenerator** I have assumed that, each iteration of the outer loop as 1 minute and every iteration of inner for loop as 1 server. This makes it easy to generate a file in quick time without having to wait for a day.
5. (Most Important)To search the log file the log data is first loaded into the program and then search operation is performed. In this case, the memory required to store the log data into the program requires large heap size. Hence, the **heap size** that I have maintained to run the program is **1024 MBs**. Heap size on the testing system should be correspondingly increased.
   * How to increase heap memory in Eclipse Neon: On Eclipse menu, clicks Run -> Run Configurations.., select the Java application you want to run, clicks on the Arguments tab, update the VM arguments with the following options  **-Xms512M or -Xmx1024M**



**Classes:**

The code uses following three classes:

1. LogGenerator: Function: Writes log info to a text file.
   1. Implements only main method.
   2. The filename to which the log is to be written should be passed as command line argument.
   3. As code writes log for only one day i.e 2016-July-09, the Unix time of 12 AM EST of this date is statically defined in the code as “t=1468036800”. And this could be modified for different dates.
   4. The class uses two for nested for loops. Outer one simulates timer. That is, every iteration of this represents minute of the day. Inner for loops represents servers (each with CPU id 0 and 1 ) i.e every iteration is simulated as server with a cpu id.
   5. The Unix time stamp of each minute, IP address, CPU id and CPU usage are written into file one after the other each separated by a tab.
   6. Total number of entries in the log file with this logic will be = 24\*60\*1000\*2, which is equal to 2,880,000.
2. LogQueryTool: Contains the main method to control basic functions such as Getting Inputs from the user, validating input, searching through the log and printing the output by creating object of class “LogSearch”.
   1. The class also implements logic of asking the user if he wants to continue querying from the log depending on user’s choice as “Y” or “N” for yes or no.
3. LogSearch: This class implements methods (mentioned in the 2nd point) which are controlled by “LogQueryTool”. The methods are explained below.
   1. loadData(TreeMap hlog, String filename): Returns TreeMap.
      1. The function takes a TreeMap and Filename which is passed as a command argument and returns TreeMap which is loaded with all the log data.
      2. Why TreeMap? : There are couple of reasons. First, you can store data as key value pairs and it gives you worst case complexity of O(1) to access anything stored as key value pairs. In our case I have chosen combination of first three columns from the log file (i.e Unix time stamp, IP address and CPU ID) as key to the TreeMap and CPU usage as Value. So that it is quicker to find CPU usage when user enters Date, IP and CPU ID. Second, as TreeMap maintains the sorted order of keys it is easier to debug when you want to see on the console if data read from the file and loaded into the program is complete and correct.
   2. getUserInput():Returns string of data entered by user as Input (Date, IP and CPU ID).
      1. Uses scanner to read data from console.
   3. validateUserInput(String userInput): Returns Boolean
      1. The function validates user input format, range and returns true of the entered data is valid else returns false.
      2. The function also prints corresponding error messages on the consol for incorrect input values.
   4. printLogInfo(TreeMap hlog, String[] userInputSplit) : A void function.
      1. Function takes the TreeMap loaded with log file data and searches for the cpu usage (value in the TreeMap) based on user input Date, IP and CPU ID (Key in the TreeMap). The function also prints the value searched onto the console.
   5. getUnixDate(String date) : Function returns Unix date of date entered by the user.

**Results:**

Code was developed/tested on 32 Bit Windows 10, 3GB RAM, Intel i5 2.4GHz system using Java in Eclipse Neon IDE.

1. Time to generate log for 1000 servers (2 CPUs each) each minute for 24 hours and write into a text file: **10 to 12 seconds**.
2. Time to load log data (2,880,000 entries) into program: **2 to 2 min 30 sec.**
3. Time to query from the log and user input validations: **Less than a second.**

**Sample Output:**

Initializing..Please Wait!

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 5:00 2016-07-09 5:05

On 2016-07-09 at 5:0 CPU usage: 4%

On 2016-07-09 at 5:1 CPU usage: 3%

On 2016-07-09 at 5:2 CPU usage: 98%

On 2016-07-09 at 5:3 CPU usage: 78%

On 2016-07-09 at 5:4 CPU usage: 87%

On 2016-07-09 at 5:5 CPU usage: 11%

Do you wish to continue (Y/N)?

Y

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 1:58 2016-07-09 2:02

On 2016-07-09 at 1:58 CPU usage: 72%

On 2016-07-09 at 1:59 CPU usage: 98%

On 2016-07-09 at 1:60 CPU usage: 67%

On 2016-07-09 at 2:0 CPU usage: 67%

On 2016-07-09 at 2:1 CPU usage: 64%

On 2016-07-09 at 2:2 CPU usage: 40%

Do you wish to continue (Y/N)?

Y

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 12:00 2016-07-09 5:05

<hh1:mm1> should be less than <hh2:mm2> for a 24 hours clock.

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.256 1 2016-07-09 5:00 2016-07-09 5:05

Enter valid range for IPs (198.162.0.0 to 198.162.3.231)

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162..1 1 2016-07-09 5:00 2016-07-09 5:05

Invalid input format.

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2/1 1 2016-07-09 5:00 2016-07-09 5:05

Invalid IP format. Please enter in x.x.x.x format

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-05 5:00 2016-07-09 5:05

Log is valid only for date 2016-07-09

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07--9 5:00 2016-07-09 5:05

Enter date in <yyyy-mm-dd> format

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 sd 5:00 2016-07-09 5:05

Enter valid number of arguments

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 500 2016-07-09 5:05

Enter time in <hh:mm> format

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 5:0:0 2016-07-09 5:05

Enter time in <hh:mm> format

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 5:00 2016-07- 5:05

Enter date in <yyyy-mm-dd> format

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 5:00 2017-07-09 5:05

Log is valid only for date 2016-07-09

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 5:00 2016-07-09 5

Enter time in <hh:mm> format

Enter IP address, CPU ID and time in the format: <ip> <cpuid> <yyyy-mm-dd> <hh:mm> <yyyy-mm-dd> <hh:mm>

198.162.2.1 1 2016-07-09 00:00 2016-07-09 00:01

On 2016-07-09 at 0:0 CPU usage: 15%

On 2016-07-09 at 0:1 CPU usage: 39%

Do you wish to continue (Y/N)?

N

Program terminated.