**Requirements**:

1. Find the maximum and minimum temperatures recorded by the network of neurons using an approach which uses the minimum number of comparisons.

**Assumptions**:

1. Input will be in a text file where each line represents a particular neurons datum which is comma separated. (Look at input.txt)
2. Input is without errors, meaning no record is incomplete.
3. When running the program, the entire path to the input file is to be provided as a command line argument.
4. The temperatureUnit for each datum will be either C or F to denote degree Celsius or degree Fahrenheit respectively. This is case sensitive. There is no Kelvin.
5. The input file should not be empty.
6. Output will contain the max and min temperatures in degree Celsius regardless of the input being in Fahrenheit or Celsius written to the console.
7. File to be compiled and then executed is MinMax.java

**Version Control approach**:

1. Git has been used for version control. The code along with this document has also been pushed to my remote repository available in the following url:

<https://github.com/DhruvPrakash/FCS-assmts/tree/master/assmt1>

1. I have committed directly to master. For this project, I have not used branches.

**Approach to design**:

The comparison approach used:

A divide and conquer approach is used for this problem.

Lets assume we have just an array of integers, where n represents the number of integers.

If n is odd, then I divide the array into (n/2) and (n/2 + 1) elements, where (n/2) and (n/2+1) are rounded down to the nearest integer.

If n is even and divisible by 4 then again I have divided it into subarrays with (n/2) and (n/2 + 1) elements.

If n is even but not divisible by 4 then I have divided into subarrays with 2k and 2k+2 elements.

When I say divide into subarrays, I do not actually cut the array down. Instead I update indices to that array to represent the array as if it has been cut down.

Optimizations used:

No. of comparisons done if there is only 1 element = 0

No. of comparisons done if there are 2 elements = 1

If there are more than 2 being joined up the chain, the number of comparisons = 2 for that particular level.

Example:

Level 1: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

(2)

Level 2: [10, 9 , 8, 7] [6, 5, 4, 3, 2, 1]

(2) (2)

Level 3: [10, 9] [8, 7] [6, 5] [4, 3, 2, 1]

1. (1) (1) (2)

Level 4: [4, 3] [2, 1]

1. (1)

Explanation:

Initially n = 10.

Divide n element array into 2 subarrays with 2k and 2k + 2 elements in level 2.

(condition used: if n is even and n%4 != 0 then divide into 2k and 2k+2)

Divide the left subarray which now has n=4 into two 2 element subarrays.

(condition used: if n is even and n%4 == 0 then divide into equal halves)

Divide the right subarray into 2 subarrays with 2k and 2k+2 elements

(condition used: if n is even and n%4 !=0 then divide into 2k and 2k+2)

Number of comparisons for each of the 2 subarrays in the left part of level 3 is 1 each

(total number comparison so far = 1 + 1 = 2)

Propagate this upward to the left half of level 2. Now 2 more comparisons needed for the left half.

(total number of comparisons so far = 2 + 2 = 4)

Level 3, right subarrays -> first one has 2 elements therefore number of comparisons for that is 1. Second one needs to be further subdivided into level 4.

(total comparisons so far = 4 + 1 = 5)

Level 4, right subarrays are now 2 elements each. Number of comparisons is now 1 for each of them.

( total comparisons so far = 5 + 1 + 1 = 7)

Propagate this to level 3 and combine to get 2 more comparisons

(total comparisons so far = 7 + 2 = 9)

Propagate this and the other subarray to level 2 to get 2 more comparisons

(total comparisons so far = 9 + 2 = 11)

Propagate the left and right part of level 2 to level 1 to get 2 more comparisons

(total comparisons so far = 11 + 2 = 13)

Therefore total number of comparisons = 13