Joseph D’Onofrio 21955039

John Baroudi 40074027

COEN 346 Programming Assignment 1 Report

The program designed to complete the task outlined in this assignment consists of two java files; the main file that handles the majority of the process and a helper file consisting of a couple of variables and methods to help with the execution of the main program.

The helper file “BulbList.java” consists of two variables; an int which is initialized to zero and is used as an array index as well as a counter for the number of burnt bulbs and an array of ints that is used to store the positions of the burnt bulbs when they are identified. The file also contains an overloaded constructor so that the size of the array can be set to the value passed to the constructor as well as three methods. The first method register(int pos) will is passed the position of a bulb that has been identified as burnt and adds it to the class’ array. The second method getList() returns the array of positions once all the burnt bulbs have been identified. The last method burntCount() returns the value of our int that was used as an array index, in this circumstance it is used to indicate the number of burnt bulbs found.

The main file “Driver.java” consists of a main() method as well as two other methods and several variables. The first method bulbScanner() is passed the array of bulbs which it traverses returning a Boolean value that indicates whether there is a burnt bulb in the array. The second method FindDefective() is outlined in detail in the flow chart on the next page. It starts by incrementing the variable that counts the number of threads. It then checks if the array length is one and if true it then checks if the bulb is burnt or not and registers the burnt bulb or returns. If the array contains more than one bulb, bulbScanner() is called; if burnt bulbs are found the array is divided in two and the two new arrays are passed recursively to FindDefective() otherwise the method returns. Once the defective bulbs have been identified the array of burnt bulbs is retrieved from the “BulbList” class and used in conjunction with the burntCount() method do determine the correct output format. The output listing all of the burnt bulbs (if any) is produced and displayed for the user. Finally, an output displaying the number of threads (that was recorded in a global variable threadCount) is produced.

Implementing threads was not particularly challenging in this assignment. It was interesting to see how the parallel threading caused the listing of the burnt bulbs to vary in order given the exact same input file but a different execution. Threading could also be used to control multiple applications concurrently. For example, you could be writing an essay on MS Word while you are listening to music on Spotify. The CPU is taking turns running these applications, but the process is so fast that it seems both processes are being run at once.

Flow Chart for Driver.java Execution

