```cpp
 1: // <Copyright Owners Albara Mehene & Sean Nishi>
 2: // regex_match example
 3: #include <boost/regex.hpp>
 4: #include <boost/date_time/gregorian/gregorian.hpp>
 5: #include <boost/date_time/posix_time/posix_time.hpp>
 6: #include <iostream>
 7: #include <string>
 8: #include <fstream>
 9:
10: using boost::gregorian::date;
11: using boost::gregorian::years;
12: using boost::gregorian::months;
13: using boost::gregorian::days;
14: using boost::gregorian::date_duration;
15: using boost::gregorian::date_period;
16: using boost::gregorian::from_simple_string;
17:
18: using boost::posix_time::ptime;
19: using boost::posix_time::hours;
20: using boost::posix_time::minutes;
21: using boost::posix_time::seconds;
22: using boost::posix_time::time_duration;
23:
24:
25: int main(int argc, char* argv[]) {
26:    if (argc != 2) {
27:      std::cout << "ERROR: input only one file" << std::endl;
28:      return -1;
29:    }
30:
31:    // open the input file
32:    std::ifstream logFile;
33:    logFile.open(argv[1]);
34:    // name of file
35:    std::string logName(argv[1]);
36:    std::string outputName = logName + ".rpt";
37:    // create the output file
38:    std::ofstream outputFile;
39:    outputFile.open(outputName.c_str());
40:    // space
41:    std::string line;
42:    date stored_date;
43:    date finished_date;
44:    ptime beginTime;
45:    ptime endTime;
46:    boost::smatch m;
47:    time_duration total_time;
48:    // space
49:    bool s_boot = false;
50:    int lineNum = 1;
51:
52:    // Start of boot: 2014-02-01 14:02:32: (log.c.166) server started
53:    boost::regex Boot_Start(
54:      "([0-9]{4})-([0-9]{2})-([0-9]{2}) "
55:      "([0-9]{2}):([0-9]{2}):([0-9]{2}): "
56:      "\\(log.c.166\\) server started.*");
57:    // If we find the text:
58:    // "2014-01-26 09:58:04.362:INFO:oejs.AbstractConnector:Started
59:    // SelectChannelConnector@0.0.0.0:9080"
60:    boost::regex Boot_End(
61:      "([0-9]{4})-([0-9]{2})-([0-9]{2}) "
```

```
 62:      "([0-9]{2}):([0-9]{2}):([0-9]{2}).([0-9]{3}):INFO:"
 63:      "oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*");
 64:    // Space
 65:    boost::regex start_service(
 66:        "Starting\\ Service\\.\\ \\ ([a-z]|[A-Z]+).+");
 67:    boost::regex end_service(
 68:        "Service\\ started\\ successfully\\.\\ \\ "
 69:        "([a-z|A-Z]+).+\\(([0-9]+).+");
 70:    // check if input file is open
 71:    if (!logFile.is_open()) {
 72:        std::cout << "ERROR: no input log file" << std::endl;
 73:        return -1;
 74:    } else {
 75:        // Go through the loop
 76:        while (getline(logFile, line)) {
 77:        // search the start regex code
 78:         if (regex_search(line, m, Boot_Start)) {
 79:            // store them into date and time
 80:            stored_date = date(stoi(m[1]), stoi(m[2]), stoi(m[3]));
 81:            beginTime = ptime(stored_date,
 82:              time_duration(stoi(m[4]), stoi(m[5]), stoi(m[6])));
 83:          // condition to see if it will fail to go to the next condition
 84:           if (s_boot) {
 85:           s_boot = false;
 86:           outputFile << "**** Incomplete boot ****\n" << std::endl;
 87:          }
 88:          // draw the start into the output file
 89:         outputFile << "=== Device boot ===\n"
 90:                  << lineNum << "(" << logName << "): "
 91:                  << m[1] << "-" << m[2] << "-" << m[3]
 92:                  << " "
 93:                  << m[4] << ":" << m[5] << ":" << m[6]
 94:                  << "Boot Start" << std::endl;
 95:          s_boot = true;
 96:        // Then do the same to the end buy checking the regex
 97:        } else if (regex_search(line, m, Boot_End)) {
 98:           finished_date = date(stoi(m[1]), stoi(m[2]), stoi(m[3]));
 99:           endTime = ptime(stored_date,
100:             time_duration(stoi(m[4]), stoi(m[5]), stoi(m[6])));
101:           // output the rest
102:           outputFile << lineNum << "(" << logName << "): "
103:                    << m[1] << "-" << m[2] << "-" << m[3]
104:                    << m[4] << ":" << m[5] << ":" << m[6]
105:                    << " Boot Completed" << std::endl;
106:        total_time = endTime - beginTime;
107:         outputFile << "    "
108:                  << "Boot Time: " << total_time.total_milliseconds()
109:                  << "ms\n" << std::endl;
110:        s_boot = false;
111:        }
112:        // increment line number to go to next line
113:        lineNum++;
114:    }
115:    // closing the files
116:    logFile.close();
117:    outputFile.close();
118:    }
119:    return 0;
120: }
```