```
    1: all: ps7a
    2:
    3: ps7a: main.o
    4:         g++ main.cpp -o ps7a -lboost_unit_test_framework -lboost_regex -g
    5:
    6: main.o: main.cpp
    7:         g++ -c main.cpp -Wall -Werror -ansi -pedantic -lboost_unit_test_fram
ework -lboost_regex -g
    8:
    9: clean:
   10:         rm *.o ps7a *~
```

```
 1: //  Copyright 2015 Zheondre Calcano
 2: //  PS7a
 3: #include <boost/regex.hpp>
 4: #include <boost/date_time.hpp>
 5: #include <exception>
 6: #include <stdexcept>
 7: #include <sstream>
 8: #include <fstream>
 9: #include <iostream>
10: #include <string>
11: #include <vector>
12:
13: using namespace std; //NOLINT
14: using namespace boost; //NOLINT
15:
16: void efname(string &name) { name += ".rpt";}
17:
18: void parse(string fn) {
19:   int linenum, completeboot;
20:   vector< int > holdval;
21:   holdval.push_back(0);
22:   holdval.push_back(0);
23:   holdval.push_back(0);
24:   string ufn, filename, lif, rs, rsa, temp, boottime;
25:   ufn = fn;
26:   efname(fn);
27:   std::fstream outfile;
28:   cout << fn << endl;
29:   outfile.open(fn.c_str(), fstream::out);
30:   rs = ".*log.c.166.*";
31:   rsa = ".*oejs.AbstractConnector:Started SelectChannelConnector.*";
32:   string t = "(\\d{2}):(\\d{2}):(\\d{2})";
33:   //  (\d{2}):(\d{2}):(\d{2})
34:   string tmm = "(\\d{2}):(\\d{2}):(\\d{2})\\.(\\d{3})";
35:   //  (\d{2}):(\d{2}):(\d{2})\.(\d{3})
36:   string gd = "(\\d{4})-(\\d{2})-(\\d{2})";
37:   boottime = "Boot Time: ";
38:   outfile << "Device Boot Repot \n" + ufn + "\n\n";
39:   std::ifstream infile(ufn.c_str());
40:   smatch sm, sn, so, sp;
41:   regex e = regex(rs);
42:   regex ea = regex(rsa);
43:   regex etime(t);
44:   regex f(tmm);
45:   regex getdate(gd);
46:   regex getdatea(gd);
47:   std::ostringstream ss;
48:   linenum = completeboot = 0;
49:   while (getline(infile, lif)) {
50:     linenum++;
51:     if (regex_match(lif, e)) {
52:       if (completeboot == 1) {
53:         outfile << "**** Incomplete boot ****\n\n";
54:         completeboot = 0;
55:       }
56:       outfile << "=== Device boot ===\n";
57:       regex_search(lif, sm, etime);
58:       regex_search(lif, so, getdate);
59:       holdval[0] = boost::lexical_cast<int>(sm[1]);
60:       holdval[1] = boost::lexical_cast<int>(sm[2]);
61:       holdval[2] = boost::lexical_cast<int>(sm[3]);
```

```
   62:       ss.str("");
   63:       ss << linenum;
   64:       temp = ss.str();
   65:       temp += "(" + ufn + "):";
   66:       temp += so[0] + " " + sm[0] + " Boot Start \n";
   67:       outfile << temp;
   68:       completeboot = 1;
   69:       temp.clear();
   70:     }
   71:     if (regex_match(lif, ea)) {
   72:       ss.str("");
   73:       ss << linenum;
   74:       temp = ss.str();
   75:       temp += "(" + ufn + "):";
   76:       regex_search(lif, sn, f);
   77:       regex_search(lif, sp, getdatea);
   78:       boost::posix_time::time_duration ta(holdval[0], holdval[1], holdval[2]
);
   79:       boost::posix_time::time_duration tb(boost::lexical_cast<int>(sn[1]),
   80:                                           boost::lexical_cast<int>(sn[2]),
   81:                                           boost::lexical_cast<int>(sn[3]));
   82:       //  tb += boost::posix_time::millisec(boost::lexical_cast<int>(sn[4]))
;
   83:       tb = tb - ta;
   84:       temp += sp[0] + " " + sn[0] + " " + "Boot Completed \n";
   85:       outfile << temp;
   86:       ss.str("");
   87:       ss << tb.total_milliseconds();
   88:       outfile <<"\t"+ boottime + ss.str() + " ms\n\n";
   89:       completeboot = 0;
   90:       temp.clear();
   91:     }
   92:   }
   93:   outfile.close();
   94: }
   95: int main(int argc, char *argv[]) {
   96:   string filename;
   97:   filename = argv[1];
   98:   if (filename.size() < 1)
   99:     throw
  100:       std::runtime_error("Null string for file name");
  101:   parse(filename);
  102:   return 0;
  103: }
```