```
    1: // Copyright 2017 Patrick Muldoon
    2: #include <vector>
    3: #include <iostream>
    4: #include <fstream>
    5: #include <string>
    6: #include "intouch.hpp"
    7:
    8: int main(int argc, char* argv[]) {
    9:         if(argc != 2) {
   10:                 throw std::runtime_error("Wrong number of arguments.");
   11:                 std::cout << "Usage " << argv[0] << "logfile" << std::end
l;
   12:                 exit(1);
   13:         }
   14:         std::string logfile = argv[1];
   15:
   16:         std::ifstream fin;
   17:         fin.open(logfile);
   18:         if(fin.fail()) {
   19:                 std::cerr << "Error opening file " << logfile << std::end
l;
   20:                 exit(1);
   21:         }
   22:
   23:         std::string outfile(argv[1]);
   24:         outfile += ".rpt";
   25:         std::ofstream fout;
   26:         fout.open(outfile.c_str());
   27:         if(fout.fail()) {
   28:                 std::cerr << "Error opening output file" << outfile << st
d::endl;
   29:                 exit(1);
   30:         }
   31:
   32:         unsigned int lines_scanned = 0, boots_started = 0, boots_finished
 = 0;
   33:         unsigned int services_started = 0, services_completed = 0;
   34:         int count = 0, counter = 0;
   35:         std::string line;
   36:         std::vector<Intouch> bootups;
   37:         std::vector<Softload> softloads;
   38:         int size = 0;
   39:
   40:         while(std::getline(fin, line)) {
   41:                 ++lines_scanned;
   42:                 if(regex_match(line, StartRegex)) {
   43:                         ++boots_started;
   44:                         Intouch it(line, logfile, lines_scanned);
   45:                         bootups.push_back(it);
   46:                         while(std::getline(fin, line) && count == 0){
   47:                                 ++lines_scanned;
   48:                                 if(regex_match(line, StartRegex)){
   49:                                         ++boots_started;
   50:                                         Intouch its(line, logfile, lines_
scanned);
   51:                                         bootups.push_back(its);
   52:                                 }else if(regex_match(line, ServiceStart))
{
   53:                                         ++services_started;
   54:                                         Services s(line, logfile, lines_s
canned);
   55:                                         bootups.back().a.push_back(s);
   56:                                 }else if(regex_match(line, ServiceSuccess
)){
   57:                                         ++services_completed;
```

```
   58:                                             std::string compare;
   59:                                             boost::smatch sm;
   60:                                             boost::regex_match(line, sm, Serv
iceSuccess);
   61:                                             compare = sm[1];
   62:                                             for(int i = 0; i < bootups.back()
.a.size(); ++i){
   63:                                                     if(bootups.back().a[i].ge
tServiceName() == compare){
   64:                                                             bootups.back().a.
at(i).ServiceBoot(line, lines_scanned);
   65:                                                     }
   66:                                             }
   67:                                     }else if(regex_match(line, SucceededRegex
)) {
   68:                                             ++boots_finished;
   69:                                             bootups.back().BootSuccess(line,
lines_scanned);
   70:                                             ++lines_scanned;
   71:                                             count++;
   72:                                             services_started = 24;
   73:                                             services_completed =24;
   74:                                     }
   75:                             }
   76:                     }else if(regex_match(line, SucceededRegex)) {
   77:                                             ++boots_finished;
   78:                                             bootups.back().BootSuccess(line,
lines_scanned);
   79:                                             ++lines_scanned;
   80:                                             count++;
   81:                     } else if(regex_match(line, SoftLoadBegin)){
   82:                             std::cout << "softload start\n";
   83:                             Softload soft(line, logfile, lines_scanned);
   84:                             softloads.push_back(soft);
   85:                             while(softloads.back().getSuccess() == false){
   86:                                     std::getline(fin, line);
   87:                                     ++lines_scanned;
   88:                                     if(regex_match(line, Original)){
   89:                                             softloads.back().Originalver(line
);
   90:                                     } else if(regex_match(line, New)){
   91:                                             softloads.back().Newver(line);
   92:                                     } else if(regex_match(line, SoftLoadEnd))
{
   93:                                             softloads.back().SoftloadSuccess(
line, lines_scanned);
   94:                                             counter++;
   95:                                     }
   96:                             }
   97:                     }
   98:
   99:                     count = 0;
  100:             }
  101:             std::cout << "here\n";
  102:             for(unsigned int j = 0; j < bootups.size()-1; ++j) {
  103:                     fout << bootups[j] << std::endl;
  104:                     fout << "Services" << std::endl;
  105:                     if(bootups[j].a.empty()){
  106:                             std::cout << "no services\n";
  107:                             fout << "There is no services due to an incomplet
e boot\n";
  108:                     }
  109:                     else{
  110:                             for(unsigned int i = 0; i < bootups[j].a.size();
++i){
```

```
  111:                                    fout << bootups[j].a[i];
  112:                            }
  113:
  114:                            fout << "\t*** Services not successfully started:
 ";
  115:                            for(unsigned int i = 0; i < bootups[j].a.size()-1
; ++i){
  116:                                    if(bootups[j].a[i].getSuccess() == false)
{
  117:                                            fout << bootups[j].a[i].getServic
eName();
  118:                                    }
  119:                            }
  120:                    }
  121:                    if(size < counter){
  122:                            if(bootups[j].getEndLine() < softloads.at(size).g
etStartLine()){
  123:                                    fout << std::endl;
  124:                                    fout << "=== Softload ===" << std::endl;
  125:                                    fout << softloads.at(size).getStartLine()
 << "(" << softloads.at(size).getFileName()
  126:                                    << ")" << " : " << softloads.at(size).get
StartTime() << " Softload Start" << std::endl;
  127:                                    fout << "\tOriginal Version ==> " << soft
loads.at(size).getOriginal() << std::endl;
  128:                                    fout << "\tNew Version ==> " << softloads
.at(size).getNew() << std::endl;
  129:                                    fout << "\tElapsed Time ==> " << std::end
l;
  130:                                    fout << softloads.at(size).getEndLine() <
< "(" << softloads.at(size).getFileName()
  131:                                    << ")" << " : " << softloads.at(size).get
EndTime() << "Softload Completed" << std::endl;
  132:                                    size++;
  133:                            }
  134:                    }
  135:                    fout << std::endl;
  136:            }
  137:            fout << bootups[bootups.size() - 1] << std::endl;
  138:            fout << "Services" << std::endl;
  139:            for(unsigned int i = 0; i < bootups[bootups.size() -1].a.size();
++i){
  140:                    fout << bootups[bootups.size()-1].a[i];
  141:            }
  142:            fout << "\t*** Services not successfully started: ";
  143:            for(unsigned int i = 0; i < bootups[bootups.size()-1].a.size(); +
+i){
  144:                    if(bootups[bootups.size()-1].a[i].getSuccess() == false){
  145:                            fout << bootups[bootups.size()-1].a[i].getService
Name();
  146:                    }
  147:            }
  148:            fin.close();
  149:            fout.close();
  150: }
```