```cpp
// Copyright 2017 Patrick Muldoon
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
#include "intouch.hpp"

int main(int argc, char* argv[]) {
        if(argc != 2) {
                throw std::runtime_error("Wrong number of arguments.");
                std::cout << "Usage " << argv[0] << "logfile" << std::endl;
                exit(1);
        }
        std::string logfile = argv[1];

        std::ifstream fin;
        fin.open(logfile);
        if(fin.fail()) {
                std::cerr << "Error opening file " << logfile << std::endl;
                exit(1);
        }

        std::string outfile(argv[1]);
        outfile += ".rpt";
        std::ofstream fout;
        fout.open(outfile.c_str());
        if(fout.fail()) {
                std::cerr << "Error opening output file" << outfile << std::endl;
                exit(1);
        }

        unsigned int lines_scanned = 0, boots_started = 0, boots_finished = 0;
        unsigned int services_started = 0, services_completed = 0;
        int count = 0, counter = 0;
        std::string line;
        std::vector<Intouch> bootups;
        std::vector<Softload> softloads;
        int size = 0;

        while(std::getline(fin, line)) {
                ++lines_scanned;
                if(regex_match(line, StartRegex)) {
                        ++boots_started;
                        Intouch it(line, logfile, lines_scanned);
                        bootups.push_back(it);
                        while(std::getline(fin, line) && count == 0){
                                ++lines_scanned;
                                if(regex_match(line, StartRegex)){
                                        ++boots_started;
                                        Intouch its(line, logfile, lines_scanned);
                                        bootups.push_back(its);
                                }else if(regex_match(line, ServiceStart)){
                                        ++services_started;
                                        Services s(line, logfile, lines_scanned);
                                        bootups.back().a.push_back(s);
                                }else if(regex_match(line, ServiceSuccess)){
                                        ++services_completed;
                                        std::string compare;
                                        boost::smatch sm;
                                        boost::regex_match(line, sm, ServiceSuccess);
                                        compare = sm[1];
                                        for(int i = 0; i < bootups.back().a.size(); ++i){
                                                if(bootups.back().a[i].getServiceName() == compare){
                                                        bootups.back().a.at(i).ServiceBoot(line, lines_scanned);
                                                }
                                        }
                                }else if(regex_match(line, SucceededRegex)) {
                                        ++boots_finished;
                                        bootups.back().BootSuccess(line, lines_scanned);
                                        ++lines_scanned;
                                        count++;
                                        services_started = 24;
                                        services_completed =24;
                                }
                        }
                }else if(regex_match(line, SucceededRegex)) {
                                        ++boots_finished;
                                        bootups.back().BootSuccess(line, lines_scanned);
                                        ++lines_scanned;
                                        count++;
                } else if(regex_match(line, SoftLoadBegin)){
                        std::cout << "softload start\n";
                        Softload soft(line, logfile, lines_scanned);
                        softloads.push_back(soft);
                        while(softloads.back().getSuccess() == false){
                                std::getline(fin, line);
                                ++lines_scanned;
                                if(regex_match(line, Original)){
                                        softloads.back().Originalver(line);
                                } else if(regex_match(line, New)){
                                        softloads.back().Newver(line);
                                } else if(regex_match(line, SoftLoadEnd)){
                                        softloads.back().SoftloadSuccess(line, lines_scanned);
                                        counter++;
                                }
                        }
                }

                count = 0;
        }
        std::cout << "here\n";
        for(unsigned int j = 0; j < bootups.size()-1; ++j) {
                fout << bootups[j] << std::endl;
                fout << "Services" << std::endl;
                if(bootups[j].a.empty()){
                        std::cout << "no services\n";
                        fout << "There is no services due to an incomplete boot\n";
                }
                else{
                        for(unsigned int i = 0; i < bootups[j].a.size(); ++i){
```

```cpp
					fout << bootups[j].a[i];
				}

				fout << "\t*** Services not successfully started: ";
				for(unsigned int i = 0; i < bootups[j].a.size()-1; ++i){
					if(bootups[j].a[i].getSuccess() == false){
						fout << bootups[j].a[i].getServiceName();
					}
				}
			}
			if(size < counter){
				if(bootups[j].getEndLine() < softloads.at(size).getStartLine()){
					fout << std::endl;
					fout << "=== Softload ===" << std::endl;
					fout << softloads.at(size).getStartLine() << "(" << softloads.at(size).getFileName()
					<< ")" << " : " << softloads.at(size).getStartTime() << " Softload Start" << std::endl;
					fout << "\tOriginal Version ==> " << softloads.at(size).getOriginal() << std::endl;
					fout << "\tNew Version ==> " << softloads.at(size).getNew() << std::endl;
					fout << "\tElapsed Time ==> " << std::endl;
					fout << softloads.at(size).getEndLine() << "(" << softloads.at(size).getFileName()
					<< ")" << " : " << softloads.at(size).getEndTime() << "Softload Completed" << std::endl;
					size++;
				}
			}
			fout << std::endl;
		}
		fout << bootups[bootups.size() - 1] << std::endl;
		fout << "Services" << std::endl;
		for(unsigned int i = 0; i < bootups[bootups.size() -1].a.size(); ++i){
			fout << bootups[bootups.size()-1].a[i];
		}
		fout << "\t*** Services not successfully started: ";
		for(unsigned int i = 0; i < bootups[bootups.size()-1].a.size(); ++i){
			if(bootups[bootups.size()-1].a[i].getSuccess() == false){
				fout << bootups[bootups.size()-1].a[i].getServiceName();
			}
		}
		fin.close();
		fout.close();
}
```