```
//  Copyright 2015 Zheondre Calcano
//  PS7b
#ifndef _Services_
#define _Services_

#include <boost/regex.hpp>
#include <boost/date_time.hpp>
#include <exception>
#include <stdexcept>
#include <sstream>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>

using namespace std; //NOLINT
using namespace boost; //NOLINT

class services{
  vector< int > start, end;
  vector< string > sname, StartLN, CompleteLN, ElapsedT;
  string startservice, GoodStart, allfs, fSM, startSoftload, EndSoftload;
  string l1, l2, l3, l4, l5;
  regex rs;
  int sofV;

 public:
  services() {
    sname.push_back("Logging");
    sname.push_back("DatabaseInitialize");
    sname.push_back("MessagingService");
    sname.push_back("HealthMonitorService");
    sname.push_back("Persistence");
    sname.push_back("ConfigurationService");
    sname.push_back("LandingPadService");
    sname.push_back("PortConfigurationService");
    sname.push_back("CacheService");
    sname.push_back("ThemingService");
    sname.push_back("StagingService");
    sname.push_back("DeviceIOService");
    sname.push_back("BellService");
    sname.push_back("GateService");
    sname.push_back("ReaderDataService");
    sname.push_back("BiometricService");
    sname.push_back("StateManager");
    sname.push_back("OfflineSmartviewService");
    sname.push_back("AVFeedbackService");
    sname.push_back("DatabaseThreads");
    sname.push_back("SoftLoadService");
    sname.push_back("WATCHDOG");
    sname.push_back("ProtocolService");
    sname.push_back("DiagnosticsService");
    startservice = ".*Starting Service.  ";
    GoodStart = "Service started successfully.  ";
    sofV = sname.size();
    fSM = "\t*** Services not successfully started: ";
    allfs = "\t*** Services not successfully started: ";
    startSoftload = ".*SOFTLOADSERVICE;Install started.*";
    EndSoftload =".*ExitValue from install command : 0.*";
    for (int i; i < 3; i++) {
      start.push_back(0);
      end.push_back(0);
    }
    for (int i = 0; i < sofV; i++) {
      StartLN.push_back("-1");
      CompleteLN.push_back("-1");
      ElapsedT.push_back("-1");
      if (i == sofV - 1) {
        allfs += sname[i] + " ";
      } else {
        allfs += sname[i]+", ";
      }
    }
    allfs += "\n";
    // rs ="\\(([0-9]{1,})\\)";
    rs = "\\(([^()]*)\\)";
  }
  void makeLsNull();
  void setNegvalues();
  void ServiceSuccess(string, int);
  void ServiceStart(string, int);
  void findOV(string);
  void findNV(string);
  bool SoftloadS(string, int, string);
  int SoftloadEnd(string, int, string);
  string Sformat(string);
  string LFS();
  string getfSM();
  string getCompleteLN(int x);
  string getStartLN(int x);
  string getElapsedT(int x);
  string getsr(int x);
  string AFail();
  string getSta();
  string getGS();
  void GetEtime();
  string getL1();
  string getL2();
  string getL3();
  string getL4();
  string getL5();
  regex getRS();
  int sz();
};
#endif
```