Test Plan: "The Happy Heart Program"

Introduction

This is an initial and probably buggy release. The Happy Heart Program is a program that monitors the heart functions of a hospital patient; Its purpose is to raise alarms if there is something wrong with a patient.

It specifically monitors the following:

- Pulse rate

- Blood pressure

- Blood oxygen level

Implementation details

This program either reads from a file or from a keyboard. Data is simulated to arrive every 10 seconds. Whenever data is read, the program prints the time, the alarm level, and a problem description if a problem exists. Bad data is considered an equipment malfunction and the program will output it as such.

Pulse

Pulse is an integer value from 0-210. There are 5 different alarm thresholds:

- Under 20 - high

- Over 170 - high

- Under 40 - medium

- Over 130 - medium

- Over 110 - low

Anything over 210 or under 0 is considered an equipment failure.

Oxygen

Oxygen is an average double value from 0-100; the average is based on a one minute moving average. There are 4 alarm levels:

- Average below 50% - high

- Average below 80% - medium

- Average below 85% - low

- 3 missed reads - low. (Equipment failure, invalidates average)

Anything under 0 or over 100 is considered an equipment failure.

Blood pressure

Blood pressure is a pair of integers separated by a '/' character. There are 4 alarm values:

- Over 150 (sys)/ 90 (dia) - low

- Over 200 (sys)/ 120 (dia) - medium

- Under 70 (sys)/ 40 (dia) - medium

- Under 230 (sys)/ 150 (dia) - high

Any values above 230 (sys)/ 150 (dia) are considered equipment failures.

Test methods

The testing for this program was a combination of black and white box testing. Because of the nature of the program specification, there are several instances of boundary value testing (at maximums, minimums, and impossible values), and equivalence class testing. The program is stateful so it was also necessary to trace the nodes of the program and test where predicates branch the program logic in several places. Redundancy is at an attempted minimum. The statefulness of the program (specifically the oxygen average) also requires several instances of data reads that should be ignored because they get the program to a testable state.

Items to be tested

Its impossible to test all values, but equivalence classing and boundary value testing allow a program to hopefully capture a good representative snapshot of all values. As stated before, values that approach a boundary (maximums, minimums, and impossible values) are going to be tested. State transitions will be tested (getting the program to a "high, "medium", "low", and "none" alarm state). These states will also be tested for each metric being read (Pulse, oxygen, and BP). The number of values being input into the program is also important (specifically for oxygen), as such, that will be tested as well.

Items not tested

The time component is specifically not being tested; its simple enough that it probably doesn't require testing, but beyond that, it will not cause a catastrophic failure if it isn't perfect. The rest of the program can determine if someone lives or dies if it isn't implemented correctly, so more of the testing focus is centered there. The risk is low with the rollover clock, but is infinitely much higher with the other values.

Goals of testing

First and foremost the purpose of these tests are to gather information about the system, qualify and quantify the system with this data, and mitigate the risk involved with the software. Hopefully the tests and subsequent analysis can emphasize robustness, reliability, and correctness. Things like security or scale aren't really specific to this program. They don't matter as much as the aforementioned values.

How to test

A list of test cases have been provided. The tests go in-depth on what is being tested and why. Use the test cases by passing them to the program in a command line interface. From the folder provided, run the following command:

$ *./term1.exe test-cases.txt*

Testing from the command line is also encouraged. If the program is run without the file argument, it will take in data from the command line.

*$ ./term1.exe*

About the tests

The tests usually follow a format of a test number, what is being tested, and what value can be expected from the test. This is the informal format:

*Test 38, systolic is at 70*

*Expect "none" status and message of "Everything is normal*

*80 90.0 70/80*

The tests will write to the command line.