Software Testing 2018

Term project

A Heart Monitor
&
A Hangman Game

Sunday, 27 May 2018

Group Iran:

Name	Student ID	Time
Ibrahim Kanj	2615072	26 hours
Chao Zhang	2619800	25 hours
Ali Nikouei	2557346	23 hours
Judith Schermer	2528697	0 hours

Task 1 - Project #1. Coder- view testing

A Heart Monitor

This program simulates the controller of a heart monitoring device. The system continuously monitors the heart functions of a hospital patient. Your monitor should read the pulse, oxygen level and blood pressure. Its main purpose is to raise alarms if there is something seriously wrong, and the patient might die. Search medical literature to determine the normal and life threatening values. Create differentiated warnings, because not all combinations of readings are equally dangerous. For example, the fact that a patient has high blood pressure is less dangerous on short term than low pulse and low blood pressure. As this is just a simulation, you will have to replace real sensors readings with simulated data

Introduction (2 hours)

What is measured? Which values are normal?

The normal pulse rate for healthy adults ranges from 60 to 100 beats per minute. Athletes may have heart rates near 40 beats per minute and experience no problems [1].
The blood pressure consists of 2 values: the systolic blood pressure and the diastolic blood pressure. If the systolic blood pressure is 140 mmHg or higher, or the diastolic blood pressure is 90 mmHg or higher, blood pressure is said to be high [2]. For people older than 70, the range of the diastolic pressure is higher: between 140-150 mmHg is still OK. A blood pressure is low, when the systolic value is lower than 90 mmHg or the diastolic is lower than 60 mmHg.

The oxygen level should fall between 80 and 100 mmHg, when measured with ABG, and between 95 and 100 percent, when measured with a pulse ox. However, in COPD or other lung diseases, these ranges might differ [3].

What is the risk? Which measurements are life threatening?

For healthy adults, it is not dangerous to have an oxygen level of 80% for a while, but a pneumonia patient, this is life threatening [4]. Therefore, the question whether particular values are life threatening, depends on the patient. However, the software system we will develop, will not take any other information about the patient as input than the pulse rate, blood pressure and oxygen level. Hence, our software system will provide warnings if the values can be dangerous(although for the particular patient, they might not be dangerous). So it is up to the doctor to handle according to the warnings or to decide that they can be ignored for this patient. We will warn the doctor when the oxygen level is below 95%. Moreover, we will warn when the pulse rate is higher than 100 or below 60.

In acute situations, it is important to get a warning when the blood pressure is too low, because then the body can not compensate anymore. Therefore, the system will give a warning when the blood pressure is low, i.e. the systolic blood pressure is lower than 90, or the diastolic blood pressure is lower than 60 [4,5]. A dangerous combination is of high pulse rate and low blood pressure.

Software Requirements Specification

(2 hours)

1) The program takes one argument which is a file with a .txt extension (i.e test.txt) containing three input values (pulse in **beats per minute**, blood pressure in **mm Hg** and oxygen level **%**)
2) The input values are separated by tabs and ends with a carriage return in case there are multiple inputs.
3) The blood pressure systolic and diastolic values are separated with a column (:)
4) Any character found in place of a number or next to a number raises a warning.
5) Example input:
   60 130:60 92
   60 150:30 90
6) Any empty file or a file with different structure than the above will raise a warning.
7) Normal Pulse is any pulse greater than or equal to 60 and less than or equal to 100. Let X be the normal pulse. X >= 60 and X <= 100.
8) Normal blood pressure values: let x be the systolic value and y be the diastolic value. Normal blood pressure scenarios:
   x>y and y >= 60 and y <=80 and x <= 120
   Or
   x>y and y< 60 and x>= 90 and x <= 120
9) Normal oxygen levels. Let x be the normal oxygen levels. x >= 95 and x <= 100
10) Any pulse P >= 0 and P < 60 is considered low pulse and any pulse P > 100 and P <= 200 is considered high pulse. If the pulse P < 0 or P > 200 then this is considered as wrong input.
11) For blood pressure values. Let x be the systolic value and y the diastolic value
    Low blood pressure scenario:
    x>y and y<60 and x < 90

    High blood pressure scenarios:
    x>y and y>80
    Or
    x>y and x > 120
12) Diastolic value is always less than the systolic value. Therefore if the input file has the following scenarios then this is considered as wrong input:
    a) Diastolic >= systolic
    b) Systolic value < 0 or > 250
    c) Diastolic value < 0 or > 140
13) Any oxygen level O >= 0 and O < 95 is considered low. If the oxygen level O < 0 or O > 100 then this is considered as wrong input.
14) **The severity of vital signs**:
    There are eight levels that explain the patient's condition:
    - Normal : when all vital signs are normal
    - Careful I : when there is a low or a high blood pressure
    - Careful II : when there is a low or high pulse
    - Careful III : when there is a low oxygen level

- Intermediate Risk I : when there is a low/high blood pressure and low/high pulse
- Intermediate Risk II : when there is a low/high blood pressure and low oxygen level
- Intermediate Risk III: when there is a low/high pulse and low oxygen level
- Maximum Risk : when there is a low/high pulse and low/high blood pressure and low oxygen level

Test Cases

(5 hours)

By looking at **100 % decisions/conditions** we come up with the following test cases.

P : represents pulse in beats per minute
OL : represents oxygen level %
SBP : represents systolic blood pressure in mm Hg
DBP : represents diastolic blood pressure in mm Hg
N : represents normal
C I : represents careful I
C II : represents careful II
C III: represents careful III
R I : represents intermediate risk I
R II : represents intermediate risk II
R III : represents intermediate risk III
MR : represents maximum risk

Test case	Inputs (test data)	Expected output	Actual Result
TC 1	P = 40	C II - Low pulse	C II - Low pulse
TC 2	P = 60	N	N
TC 3	P = 100	N	N
TC 4	P = 150	C II - High pulse	C II - High pulse
TC 5	OL = 85	C III - Low oxygen level	C III - Low oxygen level
TC 6	OL = 95	N	N
TC 7	OL = 100	N	N
TC 8	SBP = 90; DBP = 59	N	N
TC 9	SBP = 89; DBP = 59	C I - Low blood pressure	C I - Low blood pressure

TC 10	SBP = 120; DBP = 59	N	N
TC 11	SBP = 121; DBP = 59	C I - High blood pressure	C I - High blood pressure
TC 12	SBP = 90; DBP = 61	N	N
TC 13	SBP = 89; DBP = 61	N	N
TC 14	SBP = 120; DBP = 61	N	N
TC 15	SBP = 121; DBP = 61	C I - High blood pressure	C I - High blood pressure
TC 16	SBP = 120; DBP = 80	N	N
TC 17	SBP = 120; DBP = 81	C I - High blood pressure	C I - High blood pressure
TC 18	SBP = 121; DBP = 80	C I - High blood pressure	C I - High blood pressure
TC 19	SBP = 121; DBP = 81	C I - High blood pressure	C I - High blood pressure
TC 20	SBP = 89; DBP = 61	N	N
TC 21	SBP =91; DBP = 59	N	N
TC 22	SBP =91; DBP = 61	N	N
TC 23	SBP =121; DBP = 70	C I - High blood pressure	C I - High blood pressure
TC 24	SBP =119; DBP = 70	N	N
TC 25	SBP =119; DBP = 81	C I - High blood pressure	C I - High blood pressure
TC 26	SBP =119; DBP = 79	N	N

NP : normal pulse
NBP: normal blood pressure
NO : normal oxygen level
LP: low pulse
LBP: low blood pressure
LO: low oxygen
HP: high pulse
HBP: high blood pressure

Test Case	Case	Input	Expected output
TC 1	NP,NBP,NO	65 120:80 95	N
TC 2	NP,LBP,NO	65 89:59 95	C I - Low blood pressure
TC 3	NP,HBP,NO	65 121:90 95	C I - High blood pressure
TC 4	LP,NBP,NO	50 120:80 95	C II - Low pulse
TC 5	LP,LBP,NO	50 89:59 95	R I - Low pulse - Low blood pressure
TC 6	LP,HBP,NO	50 121:90 95	R I - Low pulse - High blood pressure
TC 7	HP,NBP,NO	120 120:80 95	C II - High pulse
TC 8	HP,LBP,NO	120 89:59 95	R I - High pulse - Low blood pressure
TC 9	HP,HBP,NO	120 121:90 95	R I - High pulse - High blood pressure
TC 10	NP,NBP,LO	65 120:80 80	C III - Low oxygen
TC 11	NP,LBP,LO	65 89:59 80	R II - Low oxygen - Low blood pressure
TC 12	NP,HBP,LO	65 121:90 80	R II - Low oxygen - High blood pressure
TC 13	LP,NBP,LO	50 120:80 80	R III - Low oxygen - Low pulse
TC 14	LP,LBP,LO	50 89:59 80	MR - Low oxygen - Low pulse - Low blood pressure
TC 15	LP,HBP,LO	50 121:90 80	MR - Low oxygen - Low pulse - High blood pressure

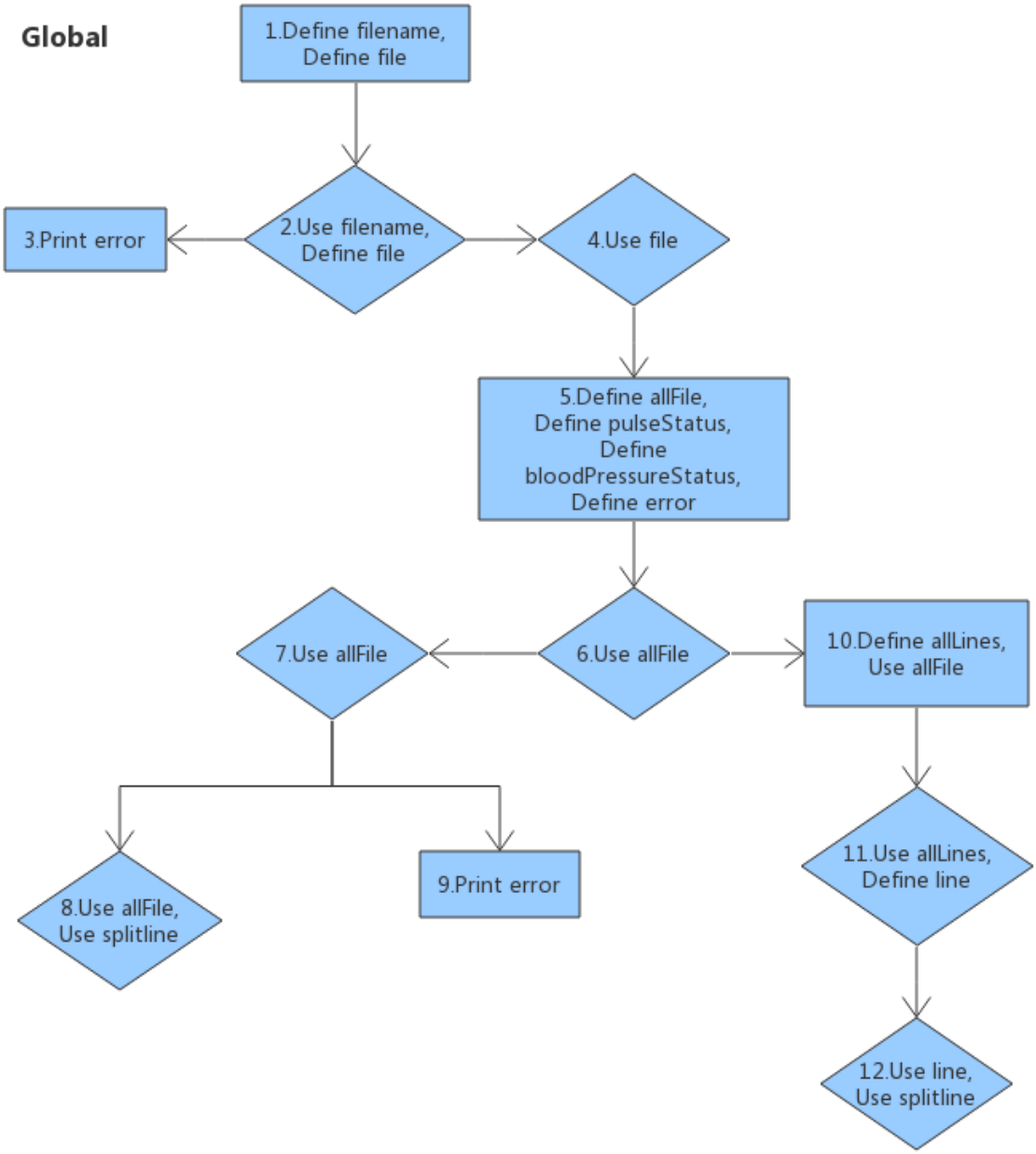
TC 16	HP,NBP,LO	120 120:80 80	R III - Low oxygen - High pulse
TC 17	HP,LBP,LO	120 89:59 80	MR - Low oxygen - High pulse - Low blood pressure
TC 18	HP,HBP,LO	120 121:90 80	MR - Low oxygen - High pulse - High blood pressure

The total coverage for the above 18 tests is 80%. The remaining things to be tested are the warnings.

Test Cases	Input	Expected output
TC 1	Pulse = 220	Pulse is not in the range
TC 2	Pulse = $	Please check your pulse input
TC 3	Blood pressure = 60:90	Systolic must be greater than diastolic
TC 4	Blood pressure = 260: 150	Systolic or diastolic outside the range
TC 5	Blood pressure = $	Please check your blood pressure input
TC 6	Blood pressure = 120/90	Please check your blood pressure input
TC 7	Oxygen = 110	Oxygen level is not in range
TC 8	Oxygen = $	Please check your oxygen input
TC 9	90 50:3 2 (spaces instead of tabs)	Please check row input
TC 10	90 50 (2 values instead of 3)	Please check row input
TC 11	No filename	Enter a file as argument
TC 12	Wrong filename	File doesn’t exist. Make sure you enter a correct one
TC 13	Empty file	File is empty

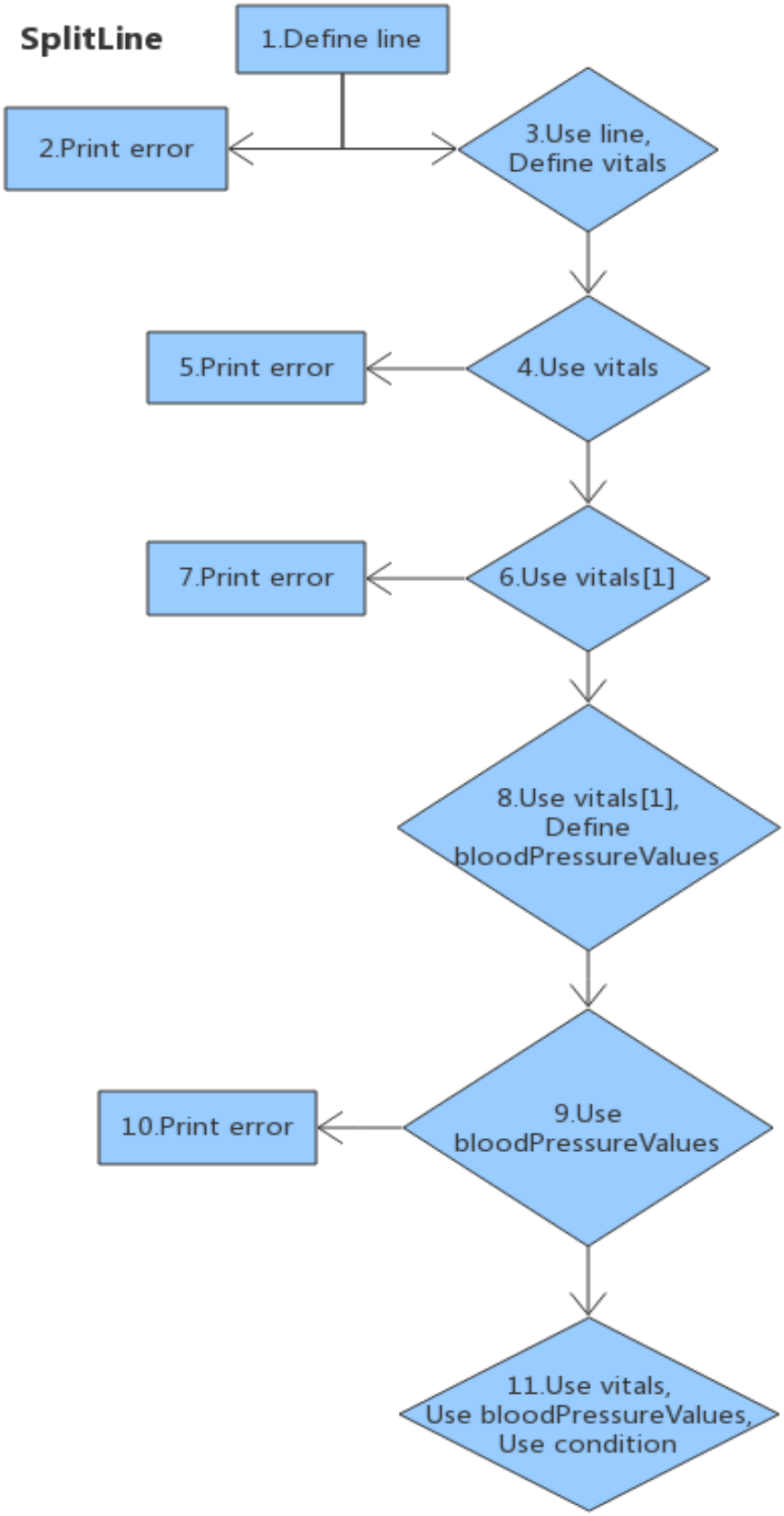
All Uses Testing

(10 hours)

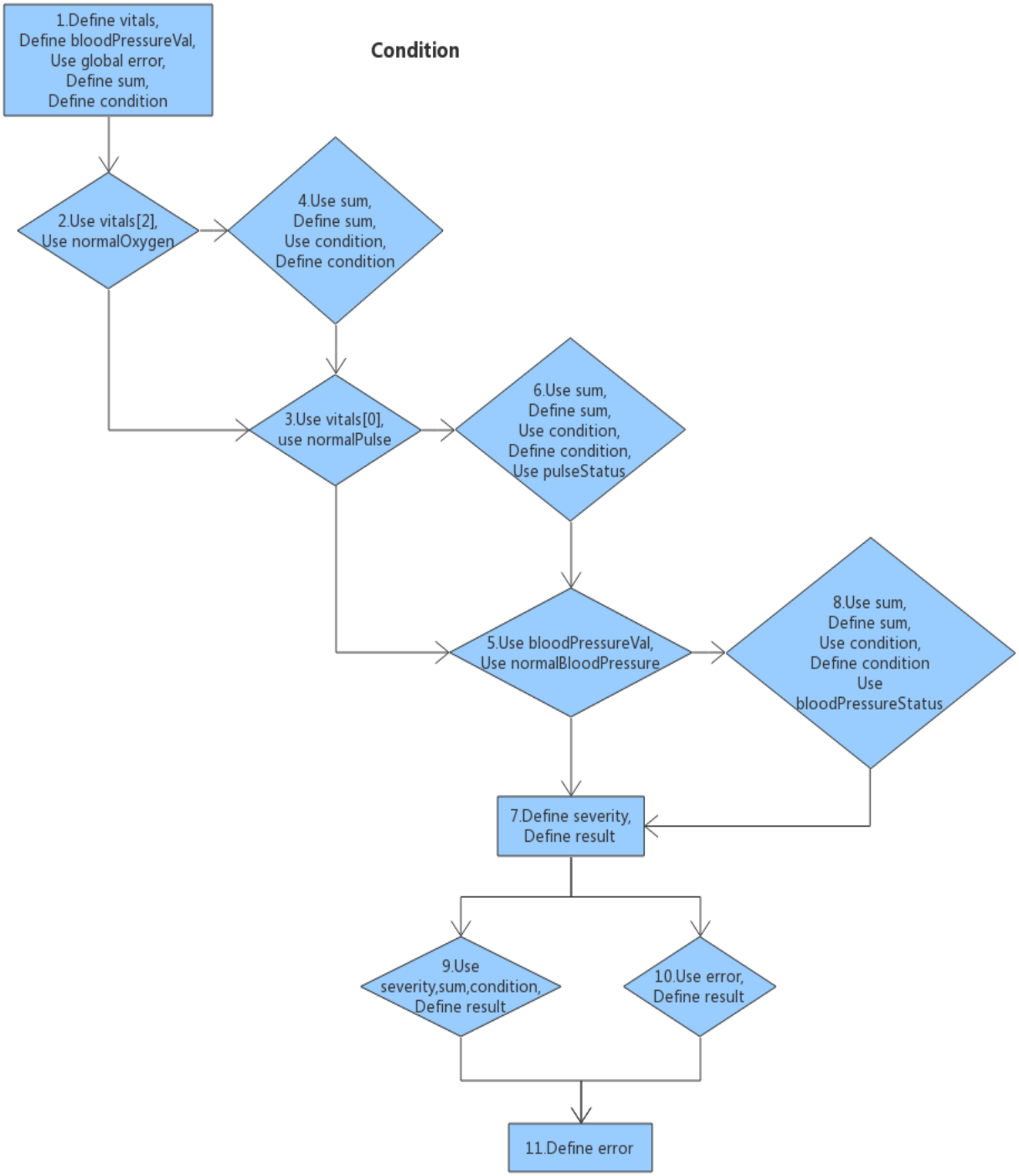


Variable	Path	Test Case	Expected Result	Actual Result
filename	1,2	“test.txt”	open this file if it exists	open file if it exists
file	1,2,4	test.txt	Read this file if it exists	Read file if it exists
allFile	5,6,10	File contains	We can split on	We split on “\n”

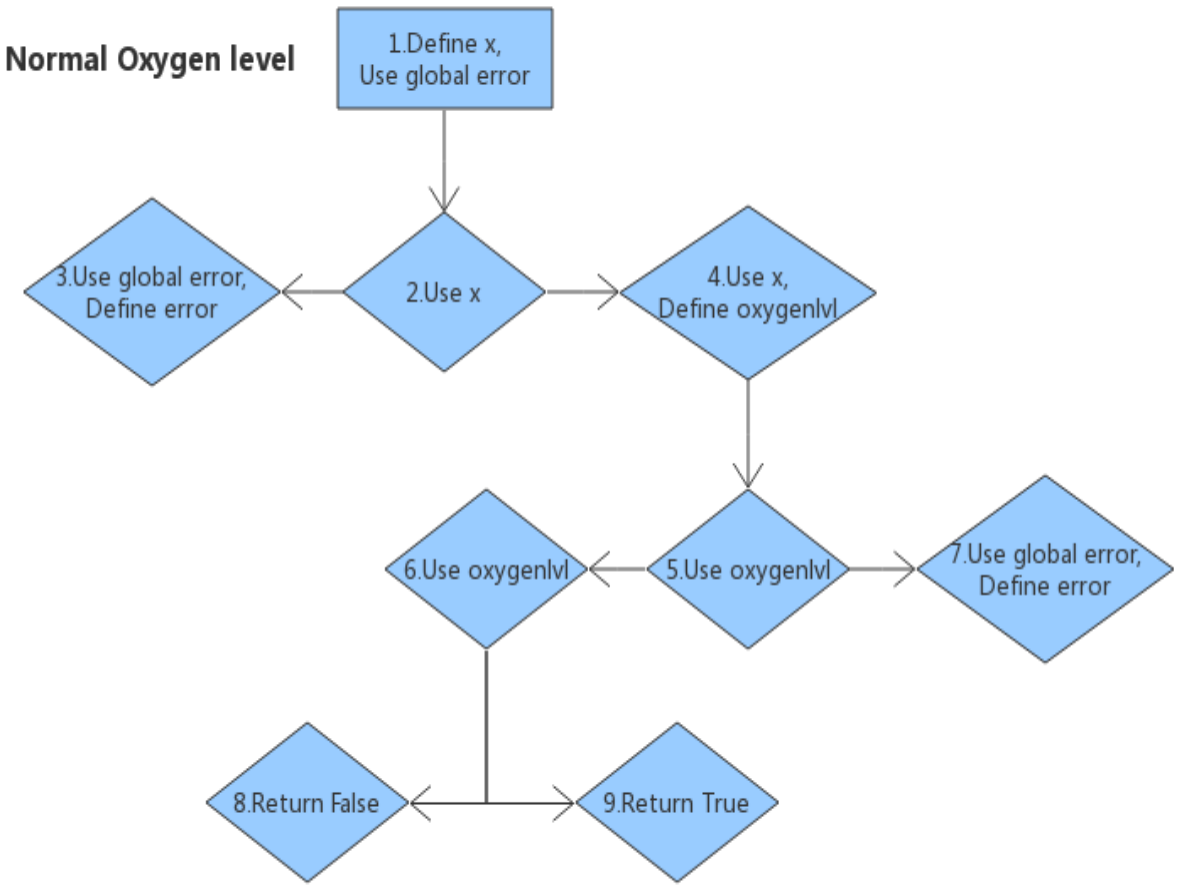
	5,6,7,8	“\n”	“\n”	
allLines	10,11	File contains \n I.e 60 120:60 97 70 120:60 97	allLines = [60 120:60 97, 70 120:60 97]	allLines = [60 120:60 97, 70 120:60 97]
line	11,12	File contains \n I.e 60 120:60 97 70 120:60 97	Line is first allLines[0] then allLines[1]	Line is first allLines[0] then allLines[1]
error	1,2,3 1,2,4,5,6,7,9	- You enter a wrong name, you have file test.txt but you write testt.txt - If test.txt is empty	* file could not be open. Make sure you entered the correct name * * caution: your file is empty *	* file could not be open. Make sure you entered the correct name * * caution: your file is empty *


SplitLine
1.Define line
2.Print error
3.Use line,
Define vitals
5.Print error
4.Use vitals
7.Print error
6.Use vitals[1]
8.Use vitals[1],
Define
bloodPressureValues
10.Print error
9.Use
bloodPressureValues
11.Use vitals,
Use bloodPressureValues,
Use condition

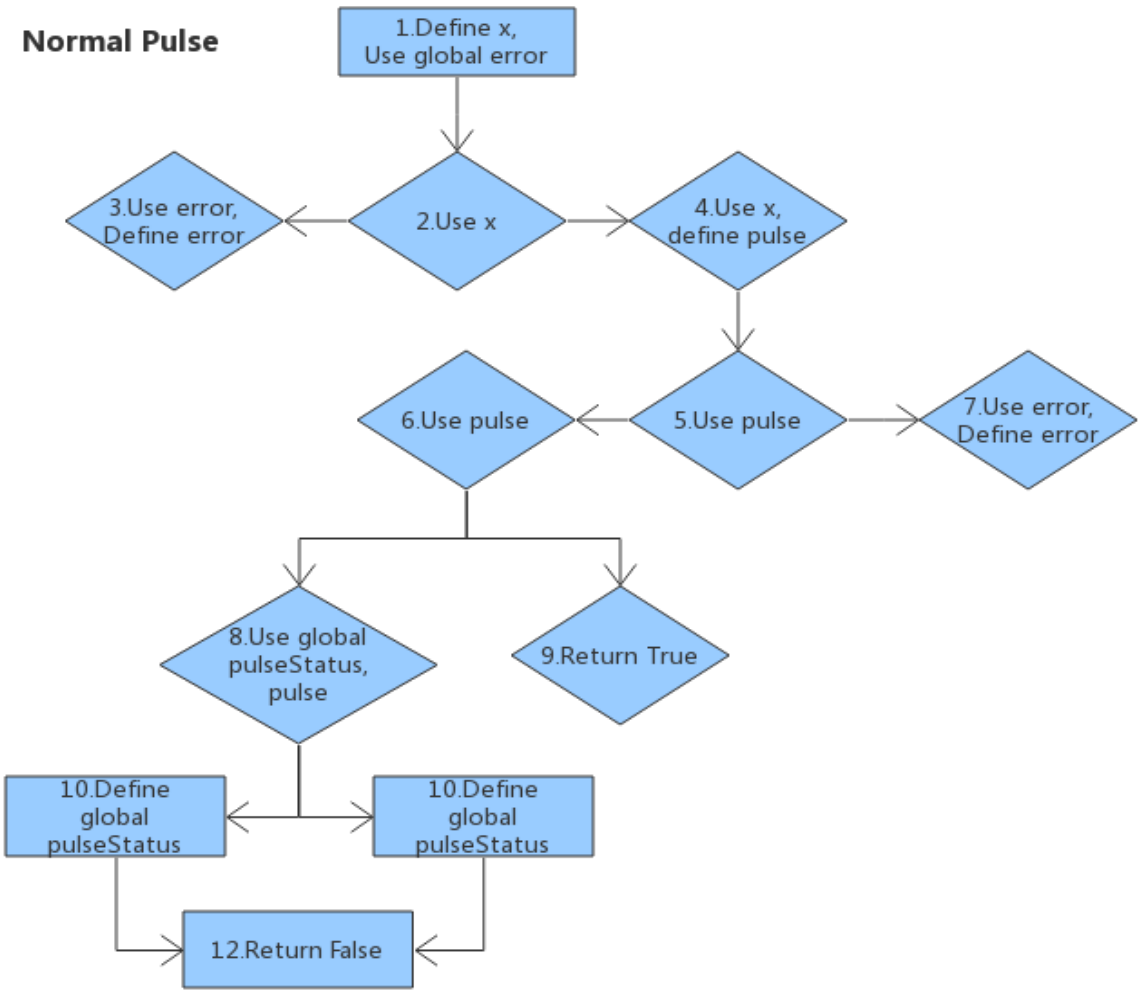
Variable	Path	Test Case	Expected Result	Actual Result
Line	1,3	Line containing tabs	Line accepted	Line accepted
Vitals	3,4,6,8,9,11	100 300:50 120	Line accepted and splitted	Line accepted and splitted
bloodPressureValues	8,9,11	100 300:50 120	Line accepted and splitted	Line accepted and splitted
error	1,2 1,3,4,5 1,3,4,6,7 1,3,4,6,8,9,10	- Line doesn't contain tabs - 100 300:50 - 100 300/60 70 - 100 300:60:70 100	* Please fix your input row * * Please fix your input row * * Please fix your blood pressure input * * Please fix your blood pressure input *	* Please fix your input row * * Please fix your input row * * Please fix your blood pressure input * * Please fix your blood pressure input *


Condition
1.Define vitals,
Define bloodPressureVal,
Use global error,
Define sum,
Define condition
2.Use vitals[2],
Use normalOxygen
4.Use sum,
Define sum,
Use condition,
Define condition
3.Use vitals[0],
use normalPulse
6.Use sum,
Define sum,
Use condition,
Define condition,
Use pulseStatus
5.Use bloodPressureVal,
Use normalBloodPressure
8.Use sum,
Define sum,
Use condition,
Define condition
Use
bloodPressureStatus
7.Define severity,
Define result
9.Use
severity,sum,condition,
Define result
10.Use error,
Define result
11.Define error

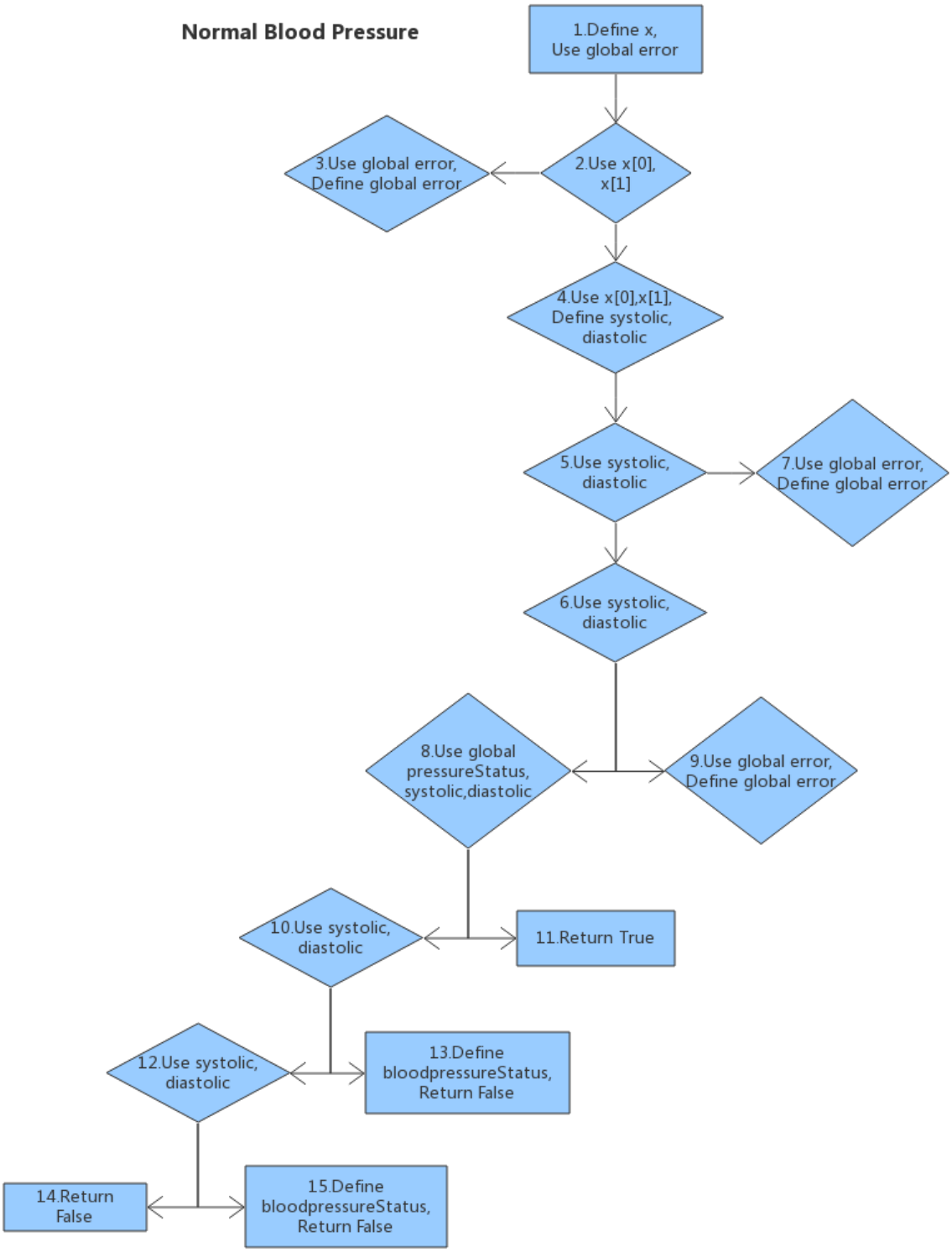
Variable	Path	Test Case	Expected Result	Actual Result
vital	1,2,4,3	60 120:60 90	Oxygen level is low	Oxygen level is low
sum	1,2,4,3,6,5,8,7,9	40 100:80 90	Sum = 24	Sum = 24
condition	1,2,4,3,6,5,8,7,9	40 100:80 90	Condition = oxygen level is low + low blood pressure + Pulse is low	Condition = oxygen level is low + low blood pressure + Pulse is low
severity	7,9	60 100:40 90	Intermediate risk III	Intermediate Risk III
result	7,9 7,10	60 120:60 97 6x 120:60 97	Normal * Please fix the error in pulse input *	Normal * Please fix the error in pulse input *
bloodPressureVal	1,2,3,5	60 120:60 97	Normal	Normal



Variable	Path	Test Case	Expected Result	Actual Result
X	1,2,4	60 120:60 90	Low oxygen	Low oxygen
error	1,2,3 1,2,4,5,7	60 120:60 9x 60 120:60 102	* Please fix the error in oxygen level input * * oxygen level is outside the range [0...100]	* Please fix the error in oxygen level input * * oxygen level is outside the range [0...100]
oxygenlvl	4,5,6	60 120:60 90	Low oxygen	Low oxygen
false	1,2,4,5,6,8	60 120:60 90	Not Normal	Not Normal
true	1,2,4,5,6,9	60 120:60 97	Normal	Normal



Variable	Path	Test Case	Expected Result	Actual Result
X	1,2,4	60 120:60 97	Normal Pulse	Normal Pulse
error	1,2,3 1,2,4,5,7	6x 120:60 97 230 120:60 97	* Please fix the error in pulse input * * pulse is outside the range [0...200] *	* Please fix the error in pulse input * * pulse is outside the range [0...200] *
pulse	4,5,6,8	60 120:60 97	Normal Pulse	Normal Pulse
pulseStatus	10, condition.6 11, condition.6	50 120:60 97 150 120:60 97	pulseStatus : low pulse pulseStatus: high pulse	pulseStatus : low pulse pulseStatus: high pulse
false	1,2,4,5,6,8,10,12 1,2,4,5,6,8,11,12	50 120:60 97 150 120:60 97	Low Pulse High Pulse	Low Pulse High Pulse
true	1,2,4,5,6,9	60 120:60 97	Normal	Normal


Normal Blood Pressure
1.Define x,
Use global error
2.Use x[0],
x[1]
3.Use global error,
Define global error
4.Use x[0],x[1],
Define systolic,
diastolic
5.Use systolic,
diastolic
7.Use global error,
Define global error
6.Use systolic,
diastolic
8.Use global
pressureStatus,
systolic,diastolic
9.Use global error,
Define global error
10.Use systolic,
diastolic
11.Return True
12.Use systolic,
diastolic
13.Define
bloodpressureStatus,
Return False
14.Return
False
15.Define
bloodpressureStatus,
Return False

Variable	Path	Test Case	Expected Result	Actual Result
X	1,2,4	60 120:60 97	Normal Blood Pressure	Normal Blood Pressure
error	1,2,3 1,2,4,5,7 1,2,4,5,6,9	60 12x:60 97 60 300:60 97 60 70:60 97	* Please fix the error in blood pressure input * * make sure systolic values is in the range [0...250] and diastolic is in the range [0...140] * * please fix your systolic and diastolic input. diastolic value should be less than the systolic value *	* Please fix the error in blood pressure input * * make sure systolic values is in the range [0...250] and diastolic is in the range [0...140] * * please fix your systolic and diastolic input. diastolic value should be less than the systolic value *
systolic	4,5,6,8,10,12	60 120:60 97	Normal Blood Pressure	Normal Blood Pressure
diastolic	4,5,6,8,10,12	60 120:60 97	Normal Blood pressure	Normal Blood Pressure
bloodPressureStatus	13, condition.8 15, condition.8	60 80:50 97 60 125:85 97	Low blood pressure High blood Pressure	Low blood pressure High blood Pressure
false	1,2,4,5,6,8,10,13 1,2,4,5,6,8,10,12,15	60 80:50 97 60 125:85 97	Low blood pressure High blood Pressure	Low blood pressure High blood Pressure
true	1,2,4,5,6,9	60 120:60 97	Normal	Normal

Agile Approach

(2 hours)

The approach we took was three phases - each phase was the equivalent of 2 days:

Phase 1: We started small and we decided to test opening, creating and reading a file and make sure that we are reading the input correctly and detecting any unwanted symbols. It is important for us to make sure that an empty file is detected and that a warning is displayed. Another thing was to test if on the command line the user is passing a filename or not.

At the end of this phase we were able to read the file and display warnings in case anything was wrong.

Errors during this phase: we were excited to get our hands dirty with the actual implementation of the requirements, so we did not cover all the faulty scenarios when handling files. Like what would happen if a file with a different extension was provided.

Phase 2: During this phase we had to code the different vital signs and what their outcome would be. So for Pulse we were interested in low, normal and high pulse. For blood pressure, low, normal and high blood pressure. For oxygen level there were only two values, low and normal. Here mutation.py was really handy and we made sure that all our test cases are returning the correct results.

Phase 3: During this phase we had to finish the program with the meaningful outcomes on the screen and this is the phase where the different vital signs would interact with one another to produce an outcome in the lines of (Normal, Careful, Intermediate risk and Maximum risk). So here we had to code and test all the 18 different combinations and make sure our expected results meets the actual results.

A total coverage of 100% in the end of this phase meant that we are covering all the statements in the code and having a 100% decision/condition coverage made us confident enough to send our code for the other group in order for them to test it.

Task 2 - Project #2. Tester-view testing

A Hangman game

This program simulates a Hangman game[6]. Basically the user has to guess a secret word by trying one letter at a time. If the letter is correct, then the computer shows the places where this letter occurs in the word; if not, then a new element is added to a hangman drawing and the user gets one step closer to the "disaster". Obviously, the number of trials is limited. The game ends when the user wins before the drawing is complete, or when the user loses and is "hung". The program should supply the user with feedback information, such as the letters he tried, the number of trials and if not guessed, also the right word.

Hangman - Requirements verification report

(8 hours)

Some Comments on the Requirements:

- Max length of a word is not mentioned. Thus now a word could be really big.
- SR2. "When no input is given, The game chooses some word to be the goal word". It doesn't actually do that when no input is given. It says a word should be minimum of 3 letters.

Hangman- Test cases based on requirements:

Here are some test cases using boundary value analysis and partitioning evaluation for the different software requirements.

SR 2)

Test Cases - acceptance of word:

Test case	Input	Expected output	Test Case Result
TC 1	abcdefghijklmnopqrstuvwxyz	should work	Successful
TC 2	ABCDEFGHIJKLMNOPQRSTUVWXYZ	should work	Successful
TC 3	ABCDefgh	should work	Successful
TC 4	x	should not work - should be min 3	Successful
TC 5	xyz	should work	Successful

TC 6	xy1	should not work - only letters	Successful
TC 7	Aa字	should not work - only letters	Successful
TC 8	no input	should not work - minimum 3	Successful

SR 3)

Word: alphabet

Test case	Input	Expected output	Test Case Result
TC 1	Guess:z	1 wrong guess	Successful
TC 2	Guess: a	a***a***	Successful
TC 3	Guess: y	2 wrong guess	Successful
TC 4	Guess: f	3 wrong guess	Successful

Previous guesses: z-a-y-f

SR 4)

Since the word is at least 3 letters, then guessing 2 letters wouldn't win one the game. similarly guessing x letters when trying to guess a word made up of n letters where x < n guessing longer words than the original could also be better given a warning and not lose a guess.

Word: ticket

Test case	Input	Expected output	Test Case Result
TC 1	guess: t	t****t	Successful
TC 2	guess: no input	It should be minimum 1	Successful
TC 3	guess: ticket	Word guessed correctly	Successful

Word: test

Test case	Input	Expected output	Test Case Result
TC 1	guess: t	t**t	Successful
TC 2	guess: a	1 wrong guesses	Successful
TC 3	guess: f	2 wrong guesses	Successful
TC 4	guess: s	t*st	Successful
TC 5	guess: j	3 wrong guesses	Successful
TC 6	guess: e	word guessed correctly	Successful

SR 5)

Word: guess

Test case	Input	Expected output	Test Case Result
TC 1	guess: g	g****	Successful
TC 2	guess: g	repeated guess - no lives losses	Successful
TC 3	guess: z	1 wrong guess	Successful
TC 4	guess: z	repeated guess - no lives losses	Successful
TC 5	guess: gifts	2 wrong guess	Successful
TC 6	guess: gifts	repeated guess - no lives losses	Successful

Previously played: g z gifts

SR 6)

Word: dictionary

Test case	Input	Expected output	Test Case Result
TC 1	guess: f	1 wrong guesses	Successful
TC 2	guess: z	2 wrong guesses	Successful
TC 3	guess: l	3 wrong guesses	Successful

TC 4	guess: a	*******a**	Successful
TC 5	guess: y	*******a*y	Successful
TC 6	guess: b	4 wrong guesses	Successful
TC 7	guess: no input	at least 1 letter	Successful
TC 8	guess: alternary	5 wrong guesses	Successful
TC 9	guess: f	repeated guess - no loves losses	Successful
TC 10	guess: alternary	repeated guess - no loves losses	Successful
TC 11	guess: s	6 wrong guesses	Successful
TC 12	guess: c	**c****a*y	Successful
TC 13	guess: m	7 wrong guesses - game lost - your word was dictionary	Successful

Previously played: f z l a y b alternary s c

Task 3 - Reflection (3 hours)

How did you select the testing techniques?

Our code has a lot of conditional statements, thus if we look at all the statements and branches then we would handle all the cases. Therefore using 100 % decision/condition coverage seemed the obvious choice. As for the hangman group that we did blackbox texting on, we tested their requirements to make sure that everything is correct and we used equivalence partitioning to check the invalid zones and the valid ones. We did not use BVA or domain testing because in case of the word, there was no maximum to detect a maximum minus 1. Moreover, there was no variables that depend on one another for domain testing to take place.

What testing you did in addition to your original test plan, and why you did it?

We decided to use all uses testing in addition to the 100% decision/condition testing because we wanted to try the technique as it is powerful and with few test cases it could prove vital. However in our case and due to the code revolving around conditional statements and no excessive defining-usage of variables in a way that could go massively wrong, all uses did not really add additional significance.

Mutation testing was another thing that proved significant in phase 2 and it would have helped a lot in phase 3 if it could work. But as the code grew and some python functions where used like {isDigit} we could not get mutation script to work again.

How realistic was your planning and how did you decide to stop testing?

Our planning was good but we were focused on coding a correct program more than writing the best requirements. So although we tested the program well, we felt that our requirements could have been a lot better and more organised and clear. As we mentioned earlier we stopped testing after having successful test cases based on 100% decision/condition coverage as - all uses testing and with 100% total code coverage.

How compares your white- box testing to the black- box test report you received, and how do you feel about this?

The white-box testing we did had around 40 test cases and it covered all the decision/condition in the code so we are sure that our code does what is intended of it to do. However the black-box testing that we received was actually really well conducted as it included all the testing techniques from equivalence partitioning, boundary value analysis as well as domain testing. Kudos to the testers that although they had a lot more than 40 test cases but they completely covered the whole program and recommended us to also check the file extension so that we accept only one file extension (.txt) as other file extensions showed a non-clear warning that does not explicitly say that the file extension is wrong.

What did you learn from all this experiment?

We learnt that system requirements specification is really important and that well written requirements could mean the difference between a fast start for developers to start coding or a slow start as to understand the intricacies hidden in the requirements. Moreover, it is an exercise of communication between the team members and agreeing when it is time to stop and move to the next phase or how many phases will there be or when should we stop testing. These were questions that are being raised all the time and finding answers to them meant that we should prioritize things and decide on our next moves and this made the experiment close to what actually happen in the field.

Conclusion

We liked the ability to test a closed system where we do not see the code. Although there is not much of techniques in black box testing but the ones available gave a flavour of what can be done as to shake the authenticity and rigidness of the program. Although it stood tall in face of our test cases but we gave it more validity that it works as intended. White box testing on the other hand is more challenging to the idea that we wrote a good code. When we were breaking a system, we were more creative with how to bring it down. If we can treat our code with the same fashion then that would be perfect. There was not really a part that we did not like, but some of the tasks took a lot of time. So mainly time is the problem in achieving more favorable results.

Appendix

1. Heart monitor code:
https://drive.google.com/open?id=1Jq4Yap1um80EodXbv6Dc9Q_aUdH-kso2
2. All combinations input file:
https://drive.google.com/open?id=1aTheuQQO9Cblx4jFkR3YwzDGTbB3Rrop

Bibliography

1. *https://www.urmc.rochester.edu/encyclopedia/content.aspx?ContentTypeID=85&ContentID=P00866*
2. https://www.hartstichting.nl/risicofactoren/gids-bloeddruk/wanneer-is-de-bloeddruk-te-hoog?tab=2
3. https://www.healthline.com/health/normal-blood-oxygen-level#oxygen-levels
4. Interview with Thomas Sierkstra (BSc) and Kyra Baelde (BSc), 5th year medical students from VU.
5. https://www.mayoclinic.org/diseases-conditions/low-blood-pressure/symptoms-causes/syc-20355465
6. https://en.wikipedia.org/wiki/Hangman_(game)