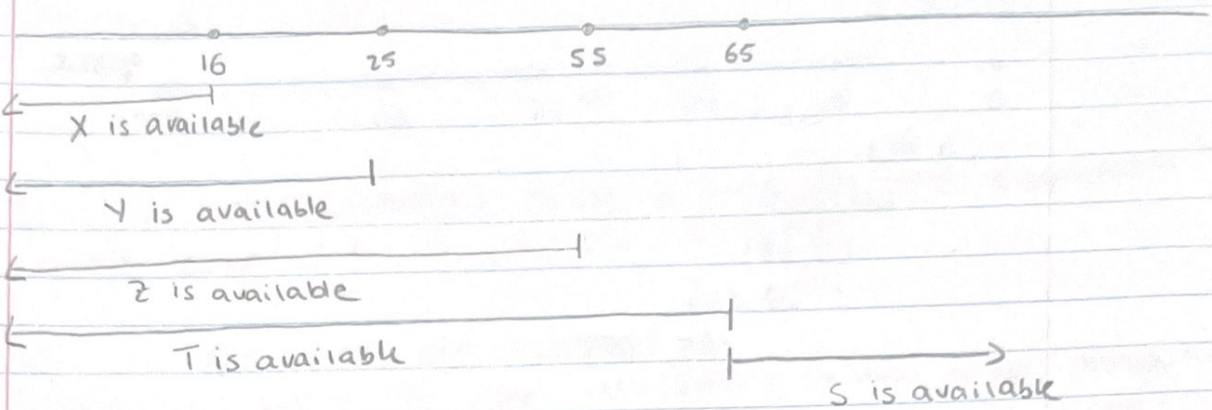
Juanita Rodelo, 101141857

1. Exercise 1



I. - Inputs: 5 inputs

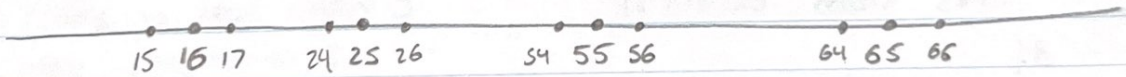
classes
① If someone is younger than 16
② If someone is younger than 25
③ If someone is below 55
④ If someone is below 65
⑤ If someone is older than 65

2. - Boundary Inputs: 4 inputs

① If someone is 16 years old
② If someone is 25 years old
③ If someone is 55 years old
④ If someone is 65 years old

∴ This leads to 9 total test input values

3.



- Adding values to my boundary inputs leads to 8 new iputs, resulting in 17 total test inputs.

Exercise 2

[0,16]
[0,25]
[0,55)
[0,65)
(65,120]

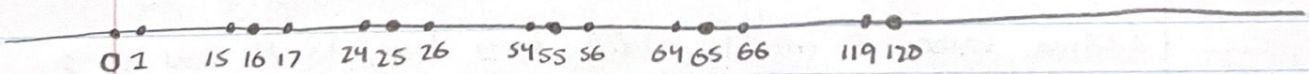
1. Inputs: 5 inputs

Classes
① If someone is younger than 16 but older than 0
② If someone is younger than 25
③ If someone is younger than 55
④ If someone is younger than 65
⑤ If someone is between 65 and 120

↳ The only thing that changed is that the first and lass class have more limitations (boundaries on both ends).

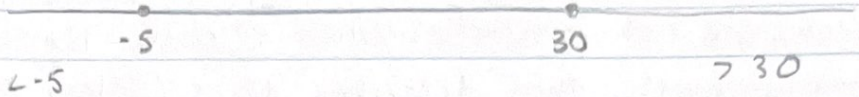
2. This adds 2 new boundary values for my classes:
① If someone is 0 years old + the ones from previous
② If someone is 120 years old. (Exercise 1, question 2)
- Resulting in 11 test input values in total (including the ones from Exercis 1)

3.



- This adds 2 new input values (including the ones from exercise 1), one close to 0 years old (1 years old) and one near 120 years old (119 years old). I did not add one on the left boundary of 0 because that's not possible. I didn't add 121 because the system does not accept that value either. Therefore, total # of inputs is 21. 3)

Exercise 3



(I) is the best selection since it hits all the equivalent classes once. ($X \leq -5$, $X \leq 30$, $X > 30$)

Why the other ones aren't good:

Ideally, we want to cover all cases in the least amount of values. Selections A, B, C, D, E, F all do not test the class where the input is equal to 5. Selections B, F, do not hit the class where the input is 30. Selections G and H do this but they have more than one input for the same class which we don't want.

Exercise 4

1.

1. Score parameter

S1: score is less than 50
S2: score is equal to 50
S3: score is greater than 50

2. Lives parameter.

L1: lives is less than 3
L2: lives is equal to 3
L3: lives is greater than 3

3. (S1 and L1) (S2 and L1) (S3 and L1)
(S1 and L2) (S2 and L2) (S3 and L2)
(S1 and L3) (S2 and L3) (S3 and L3)

∴ There are 9 combinations

4. (score = 20, # lives = 1) (score = 50, # lives = 2) (score = 60, # lives = 0)
(score = 10, # lives = 3) (score = 50, # lives = 3) (score = 100, # lives = 3)
(score = 5, # lives = 4) (score = 50, # lives = 10) (score = 200, # lives, 9)

Exercise 5

5. Month parameter:

- The blocks are not complete. When month = 12 there is no characteristic that describes this (unless Month 0 is the first and month 11 is the last)

Day parameter:

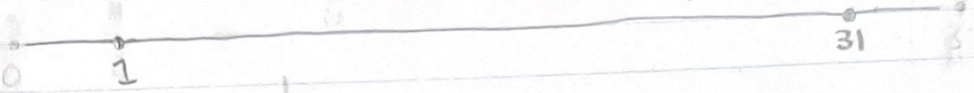
- The blocks are not disjoint. When Day = 1, both Day 2 and Day 3 cover this case.

- Overall, I think the engineer is missing various characteristics.

Exercise 6

Paramaters:

- Day:



↳ 5 equivalent classes:

① Day < 1
② Day = 1
③ 1 < Day < 31
④ Day = 31
⑤ Day > 31

- Month:

↳ 5 equivalent classes:

① Month < 1
② Month = 1
③ 1 < Month < 12
④ Month = 12
⑤ Month > 12

- Year:

* Assumming we are starting at 0 and going until infinity

↳ 3 equivalent classes

① Year < 0
② Year = 0
③ Year > 0

* This also assumes we only have to check the validity of the digit provided for each parameter.