SoftServe | Empowering your Business through Software Development

Boundary Value Testing Technique Training

Yanina Hladkova

Agenda

1. Introduction
2. Technique
3. Examples
4. Applicability and Limitations
5. Summary
6. Practice
7. References

THE BEGINNING

OF ALL THINGS

Introduction

Equivalence class testing is the most basic test design technique. It helps testers choose a small subset of possible test cases while maintaining reasonable coverage.

Equivalence class testing has a second benefit. It leads us to the idea of boundary value testing, the second key test design technique to be presented.

Introduction

What is boundary value testing?
Why do we need it?

Boundary value testing focuses on the boundaries simply because that is where so many defects hide.

Introduction: Situation 1

We are writing a module for a human resources system that decides how we should process employment applications based on a person's age. Our organization's rules are:

0-16 – Don't hire

16-18 – Can hire on a part-time basis only

18-55 – Can hire as a full-time employee

55-99 – Don't hire

Introduction: Edge

Notice the problem at the boundaries – the "edges" of each class. The age "16" is included in two different equivalence classes (as are 18 and 55).

The first rule says don't hire a 16-year-old. The second rule says a 16-year-old can be hired on a part-time basis.

Introduction: Solution 1

```
If (applicantAge >= 0 && applicantAge <=16)
      hireStatus="NO";
If (applicantAge >= 16 && applicantAge <=18)
      hireStatus="PART";
If (applicantAge >= 18 && applicantAge <=55)
      hireStatus="FULL";
If (applicantAge >= 55 && applicantAge <=99)
      hireStatus="NO";
```

Of course, the mistake that programmers make is coding inequality tests improperly.

Writing > (greater than) instead of **≥** (greater than or equal) is an example.

The interesting values on or near the boundaries in this example are:

{–1, 0, 1}, {15, 16, 17}, {17, 18, 19}, {54, 55, 56}, and {98, 99, 100}.

Introduction: Inspection

The most efficient way of finding such defects, either in the requirements or the code, is through inspection.

However, no matter how effective our inspections, we will want to test the code to verify its correctness.

Introduction: Situation 2

Perhaps this is what our organization meant:

0–15 – Don't hire

16–17 – Can hire on a part-time basis only

18–54 – Can hire as a full-time employee

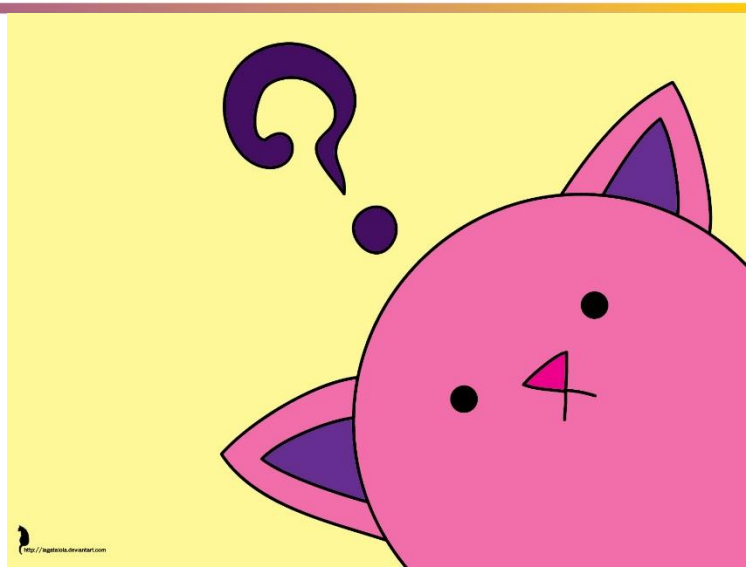
55–99 – Don't hire

What about ages -3 and 101? Note that the requirements do not specify how these values should be treated. We could guess but "guessing the requirements" is not an acceptable practice.

Introduction: Solution 2

The code that implements the corrected rules is:

```
If (applicantAge >= 0 && applicantAge <=15)
      hireStatus="NO";
If (applicantAge >= 16 && applicantAge <=17)
      hireStatus="PART";
If (applicantAge >= 18 && applicantAge <=54)
      hireStatus="FULL";
If (applicantAge >= 55 && applicantAge <=99)
      hireStatus="NO";
```



The interesting values on or near the boundaries in this example are:

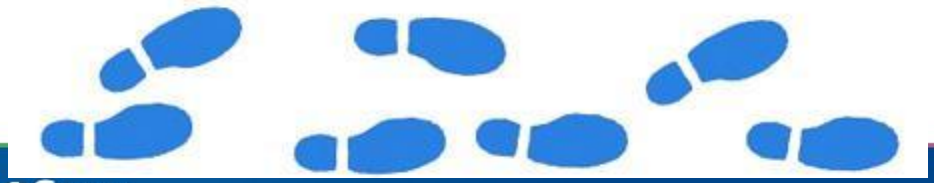
{-1, 0, 1}, {14, 15, 16}, {15, 16, 17}, {17, 18, 19}, {53, 54, 55}, {54, 55, 56}, and {98, 99, 100}. Other values, such as {-42, 1001, FRED, %$#@} might be included depending on the module's documented preconditions.

Boundary values: -1, 0, 15, 16, 17, 18, 54, 55, 99, 100.

Technique

Technique: Steps

1. Identify the equivalence classes.
2. Identify the boundaries of each equivalence class.
3. Create test cases for each boundary value by choosing one point on the boundary, one point just below the boundary, and one point just above the boundary.



Technique: Examples

"Below" and "above" are relative terms and depend on the data value's units.

If the boundary is 16 and the unit is "integer"

then the "below" point is 15 and the "above" point is 17.

Technique: Examples

If the boundary is $5.00 and the unit is "US dollars and cents"

then the below point is $4.99 and the above point is $5.01.

On the other hand, if the value is $5 and the unit is "US dollars"

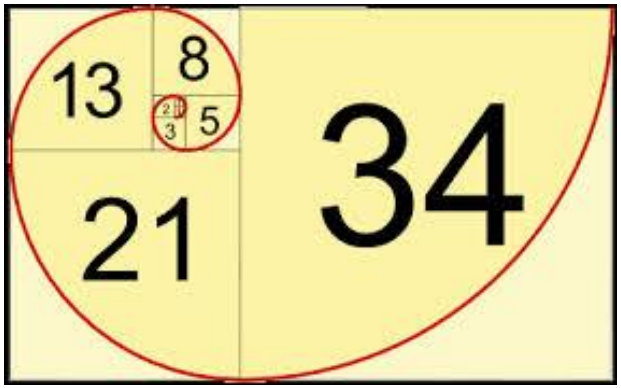
then the below point is $4 and the above point is $6.

What if user can enter anything in the field as there's no UI limitation? How will we define boundary values?

$4,(9); 5 and 5,(0)1

Technique: Examples

Which boundary values will we use if we test casino, coffee machine or Fibonacci numbers system?



Technique: Examples

Why do we need to test one value above and one value below the boundary?

x>=2

Test: 1, 2 – everything is OK

x>2

Test: 1, 2 – you will find a defect. But do you now why it happens? Which test you will do next? 10, 999, 5 or 3?

x=2

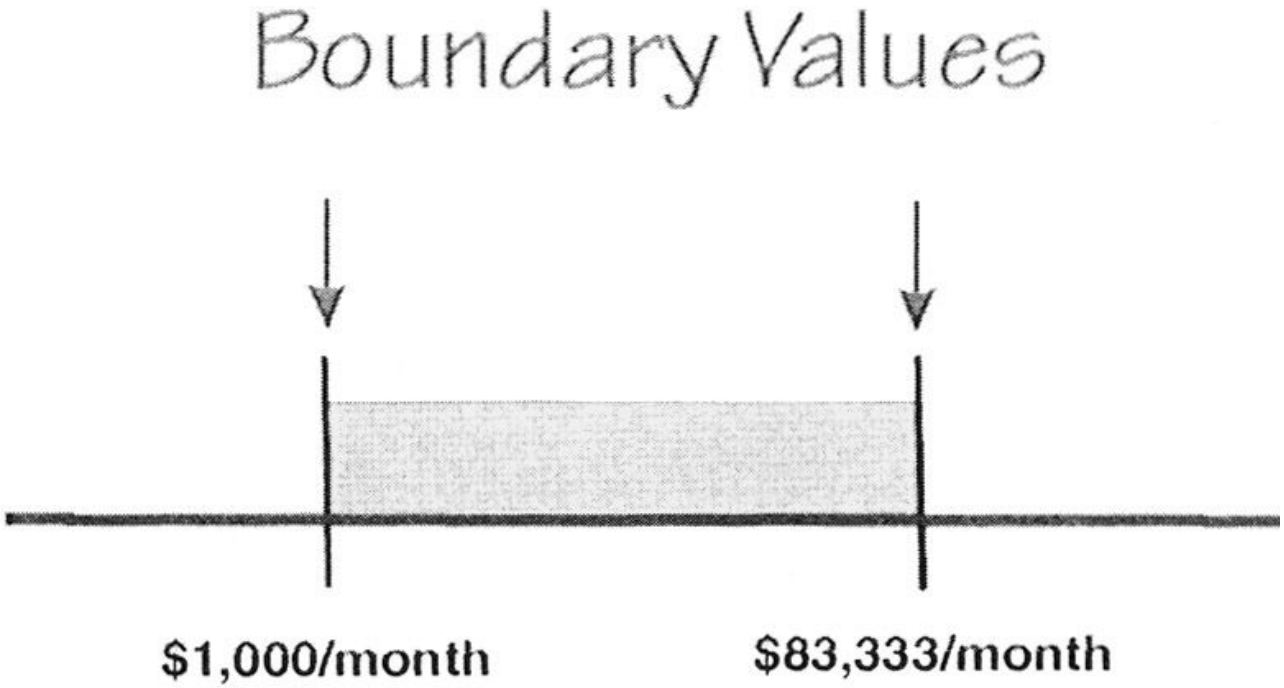
Test: 1, 2 – gives the same result but you won't find a defect, but 1, 2, 3 will gain confidence and find a defect

Technique: Tips

Note that a point just above one boundary may be in another equivalence class. There is no reason to duplicate the test. The same may be true of the point just below the boundary.

You could, of course, create additional test cases farther from the boundaries (within equivalence classes) if you have the resources. These additional test cases may make you feel warm and fuzzy, but they rarely discover additional defects.

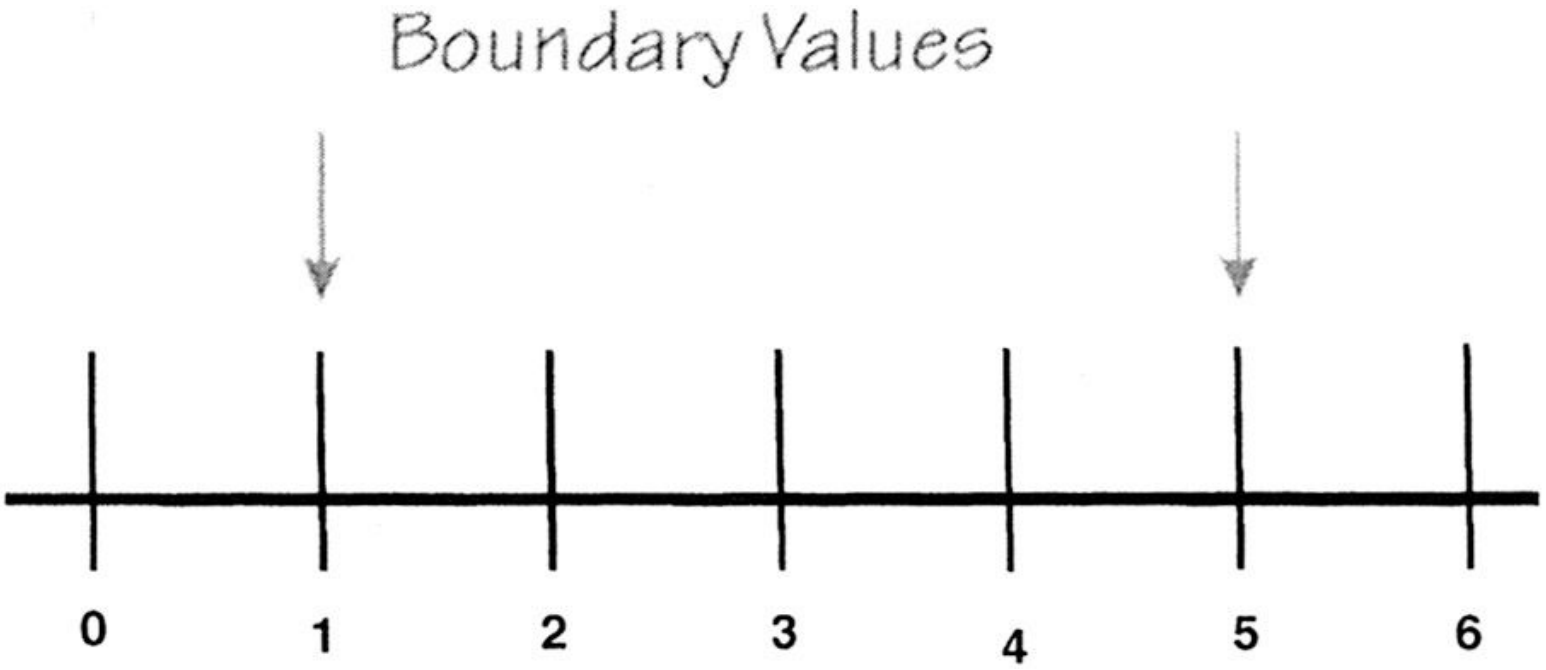
Technique: Continuous



Boundary values for a continuous range of inputs.

Test data input of {$999, $1,000, $1,001} on the low end and {$83,332, $83,333, $83,334} on the high end.

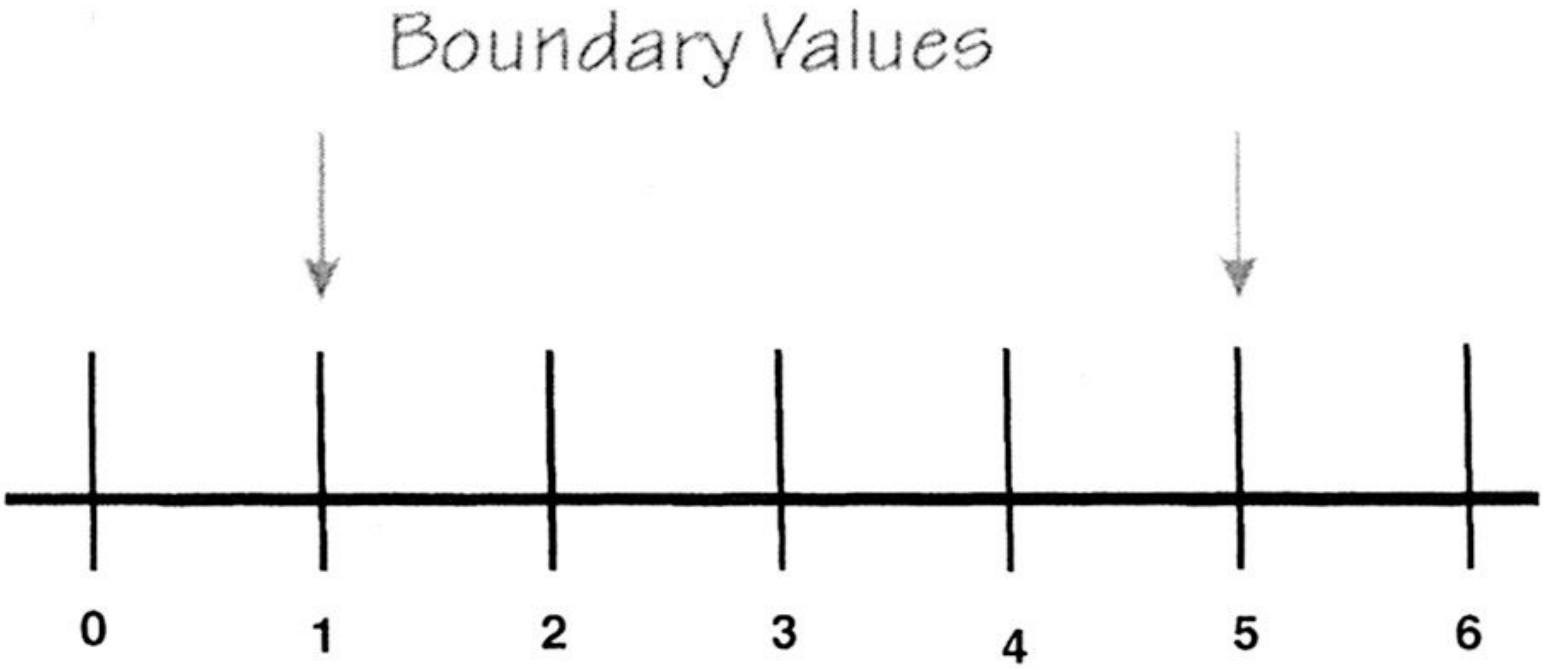
Technique: Discrete



Boundary values for a discrete range of inputs.

Test data input of {0, 1, 2} on the low end and {4, 5, 6} on the high end.

Technique: Fractional



What are boundary values here if we may have fractional numbers? Do we have points below and above the boundary?

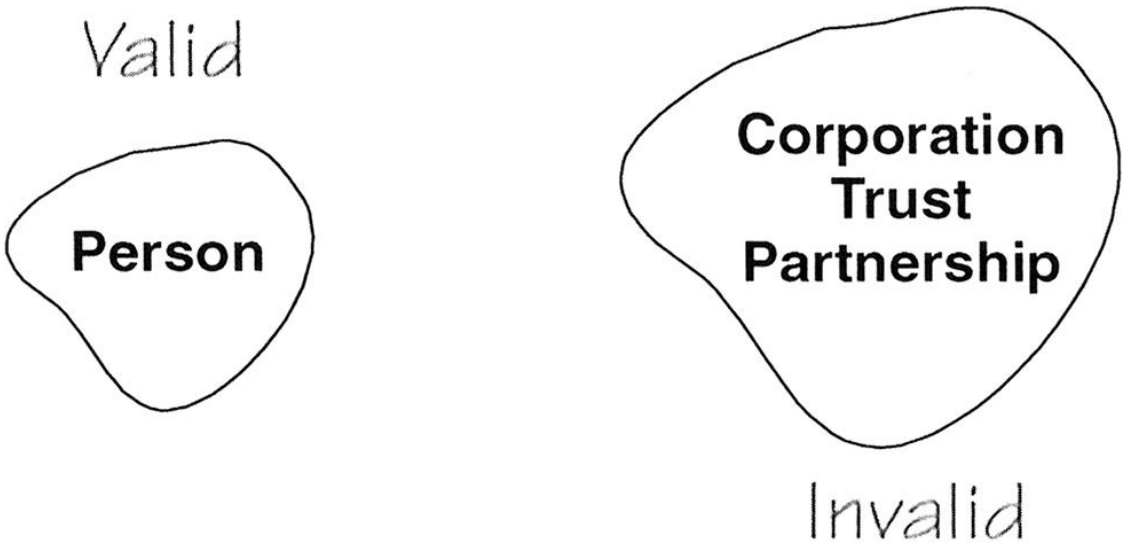
Test data input of {0,(9); 1; 1,(0)1} on the low end and {4,(9); 5; 5,(0)1} on the high end. But no exact points.

Technique: Fractional

We need to have here several solutions what to do in such cases:

1) Use only edge value for testing, don't use above\below edge values
2) Get UI requirements\limitations for this field – 4.99 is closest to 5 from UI point of view
3) Get business requirements\limitations for this field
4) Find out closest above\below values to the identified edge

Technique: Array



What are boundary values here?

No boundaries.

Technique: Combination

Rarely we will have the time to create individual tests for every boundary value of every input value that enters our system. More often, we will create test cases that test a number of input fields simultaneously.

A set of test cases containing combinations of valid (on the boundary) values and invalid (off the boundary) points.

Technique: Combination

Monthly Income	Number of Dwellings	Result	Description
$1,000	1	Valid	Min income, min dwellings
$83,333	1	Valid	Max income, min dwellings
$1,000	5	Valid	Min income, max dwellings
$83,333	5	Valid	Max income, max dwellings
$1,000	**0**	Invalid	Min income, below min dwellings
$1,000	**6**	Invalid	Min income, above max dwellings
$83,333	**0**	Invalid	Max income, below min dwellings
$83,333	**6**	Invalid	Max income, above max dwellings
$999	1	Invalid	Below min income, min dwellings
$83,334	1	Invalid	Above max income, min dwellings
$999	5	Invalid	Below min income, max dwellings
$83,334	5	Invalid	Above max income, max dwellings

Technique: Usage

Why do we need values above and below edge?

1) Higher test coverage
2) ECP replacement
3) Time & Budget constraints
4) Quicker defect location identification
5) High quality requirements for the project from Customer\Domain

Why don't we need\want to use below\above edge values?

1) Time & Budget constraints
2) Less probability to discover defects
3) Good enough test coverage for project

Examples

Examples: 1

Quantity

Input to this field can be between one and four numeric characters (0, 1, …, 8, 9).

A set of boundary value test cases for the length attribute would be {0, 1, 4, 5} numeric characters.

Examples: 2

A thermometer can have values between 34,5 and 42.

34,4; 34,5; 42; 42,1

34,4(9); 34,5; 42; 42,(0)1

Applicability and Limitations

Applicability and Limitations

- Boundary value testing can significantly reduce the number of test cases that must be created and executed. It is most suited to systems in which much of the input data takes on values within ranges or within sets.
- Boundary value testing is equally applicable at the unit, integration, system, and acceptance test levels. All it requires are inputs that can be partitioned and boundaries that can be identified based on the system's requirements.

Summary

Summary

- While equivalence class testing is useful, its greatest contribution is to lead us to boundary value testing.
- Boundary value testing is a technique used to reduce the number of test cases to a manageable size while still maintaining reasonable coverage.
- Boundary value testing focuses on the boundaries because that is where so many defects hide. Experienced testers have encountered this situation many times. Inexperienced testers may have an intuitive feel that mistakes will occur most often at the boundaries.
- Create test cases for each boundary value by choosing one point on the boundary, one point just below the boundary, and one point just above the boundary. "Below" and "above" are relative terms and depend on the data value's units.

Practice

Practice

- ZIP Code – five numeric digits. Legitimate ZIP Codes in the United States.
- Last Name – one through fifteen characters (including alphabetic characters, periods, hyphens, apostrophes, spaces, and numbers).
- User ID – eight characters at least two of which are not alphabetic (numeric, special).
- Course ID – three alpha characters representing the department followed by a six-digit integer which is the unique course identification number. The possible departments are:
  - PHY - Physics
  - EGR - Engineering
  - ENG - English
  - LAN - Foreign languages
  - CHM - Chemistry
  - MAT - Mathematics

Practice: Answers 1

- ZIP Code – five numeric digits. Legitimate ZIP Codes in the United States.

Length	Legitimate
4, 5, 6	01000, 01001, 99929, 99930
	http://www.city-data.com/zipDir.html

Practice: Answers 2

- Last Name – one through fifteen characters (including alphabetic characters, periods, hyphens, apostrophes, spaces, and numbers).

Length

0, 1, 15, 16

Example	Result	Comment
Co.- 1”qwoptyBd	Valid	Length max
G	Valid	Length min
	Invalid	Length < min
ABCDEFghijklmnop	Invalid	Length > max

Practice: Answers 3

- User ID – eight characters at least two of which are not alphabetic (numeric, special).

Length

7, 8, 9

Number of numeric and special characters

1, 2, 8, 9

Example	Result	Comment
1!abcDYZ	Valid	Length, number min
@#$%^.,)	Valid	Length, number max
1!abcDY	Invalid	Length < min, number min
0#?(cyzag	Invalid	Length > max
abcptu6w	Invalid	Number < min
“(/\,.123	Invalid	Number > max, length > max

Practice: Answers 4

- Course ID – three alpha characters representing the department followed by a six-digit integer which is the unique course identification number. The possible departments are:
  - PHY – Physics
  - EGR – Engineering
  - ENG – English
  - LAN – Foreign languages
  - CHM – Chemistry
  - MAT – Mathematics
  - PED – Physical education
  - SOC – Sociology

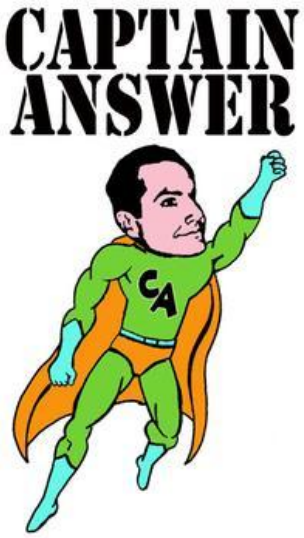
Length Alpha Characters position Length Digit Length General

2, 3, 4 first 3, 2d and 3d and 4th 5, 6, 7 8, 9, 10

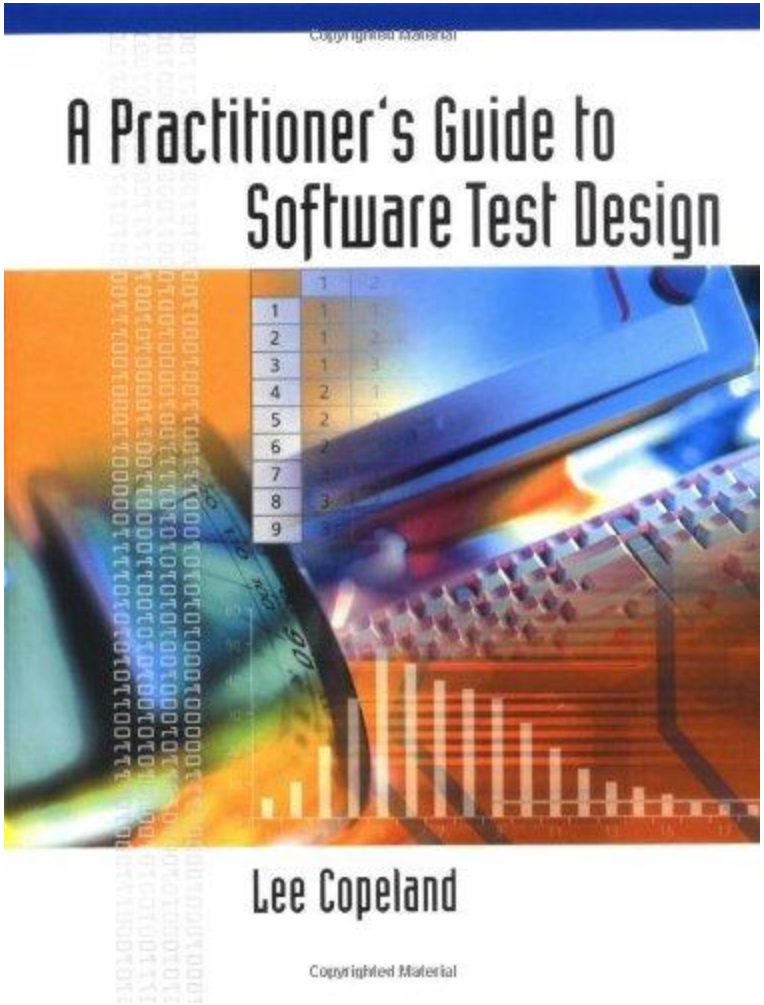
Example	Result	Comment
PHY123456	Valid	Length, position
EG9876541	Invalid	Length Alpha < min, Length Digit > max
EN987654	Invalid	Length Alpha < min, Length General < min
LAND12345	Invalid	Length Alpha > max, Length Digit < min
CHMQ345678	Invalid	Length Alpha > max, Length General > max
1MAT36974	Invalid	Characters
PED12345	Invalid	Length Digit < min, Length General < min
SOC6987451	Invalid	Length Digit > max, Length General > max

Practice: Answers 5

N	ZIP Code	Last Name	User ID	Course ID	Result
1	01001	Co.- 1"qwoptyBd	1!abcDYZ	PHY123456	Valid
2	99929	G	@#$%^.,)	CHM997410	Invalid
3	01000	[any valid]	[any valid]	[any valid]	Invalid
4	99930	[any valid]	[any valid]	[any valid]	Invalid
5	0174	[any valid]	[any valid]	[any valid]	Invalid
6	[any valid]		[any valid]	[any valid]	Invalid
7	[any valid]	ABCDEFghijklmnop	[any valid]	[any valid]	Invalid
8	[any valid]	[any valid]	1!abcDY	[any valid]	Invalid
9	[any valid]	[any valid]	0#?(cyzag	[any valid]	Invalid
10	[any valid]	[any valid]	abcptu6w	[any valid]	Invalid
11	[any valid]	[any valid]	"(/\,.123	[any valid]	Invalid
12	[any valid]	[any valid]	[any valid]	EG9876541	Invalid
13	[any valid]	[any valid]	[any valid]	EN987654	Invalid
14	[any valid]	[any valid]	[any valid]	LAND12345	Invalid
15	[any valid]	[any valid]	[any valid]	CHMQ345678	Invalid
16	[any valid]	[any valid]	[any valid]	1MAT36974	Invalid
17	[any valid]	[any valid]	[any valid]	PED12345	Invalid
18	[any valid]	[any valid]	[any valid]	SOC6987451	Invalid



References


Copyrighted Material
A Practitioner's Guide to Software Test Design
Lee Copeland
Copyrighted Material


That's all Folks!
Any Question ?


THANK
YOU


Have a nice day!