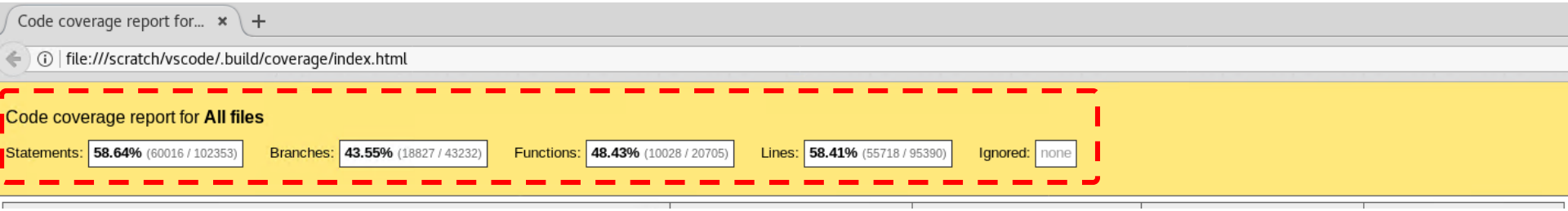
Code Coverage Recap

Example Code Coverage for Visual Studio Code

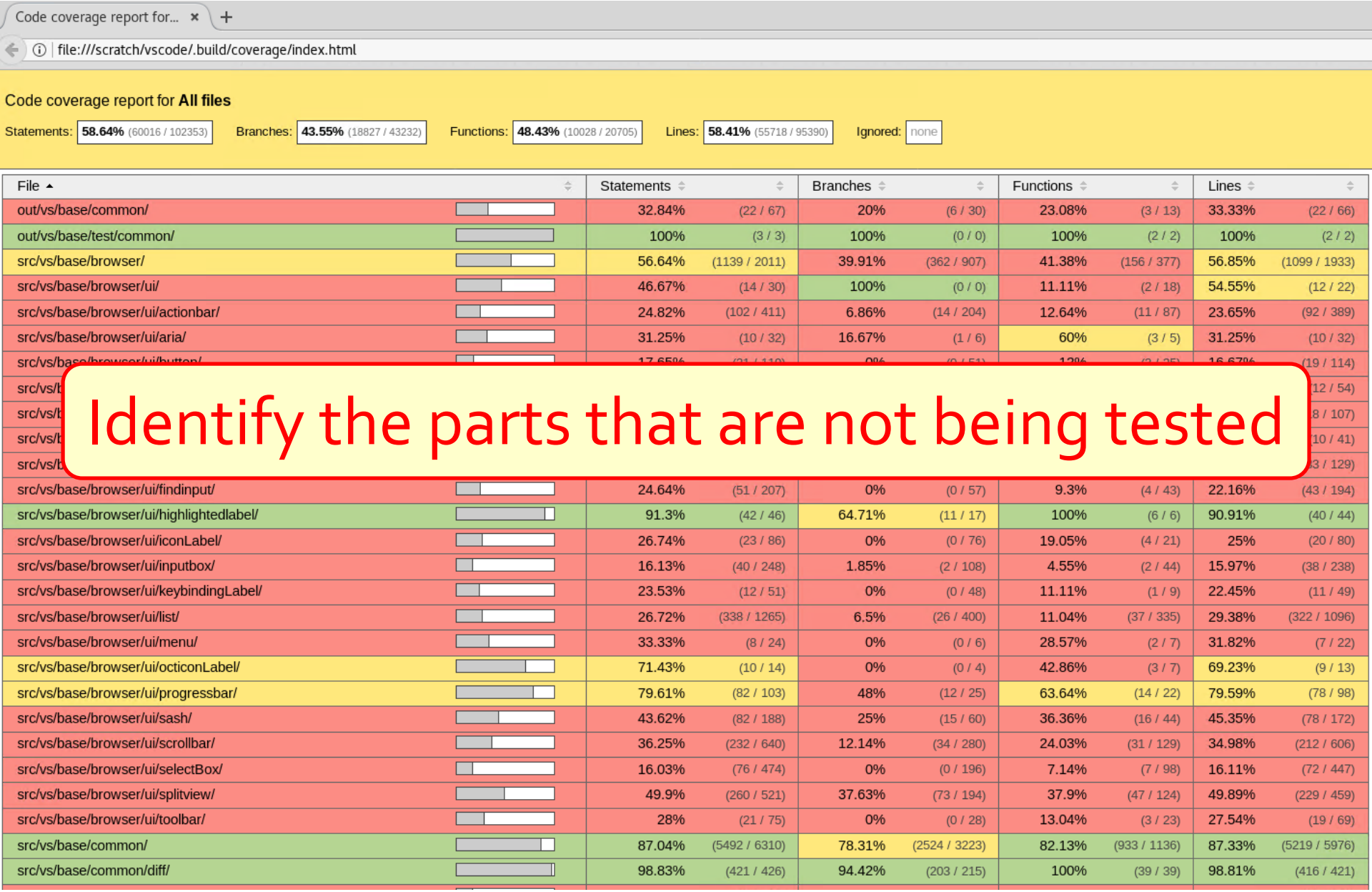


Overall coverage

- Statements
- Branches
- Functions
- Lines

https://github.com/Microsoft/vscode/wiki/How-to-Contribute#build-and-run-from-source ⇒ https://github.com/Microsoft/vscode/blob/master/test/README.md

Example Code Coverage for Visual Studio Code



https://github.com/Microsoft/vscode/wiki/How-to-Contribute#build-and-run-from-source ⇒ https://github.com/Microsoft/vscode/blob/master/test/README.md

Beyond Functional Testing

- Code Coverage
- Equivalence Partitioning
- Boundary Value Analysis

Equivalence Partitioning Motivation

- Test suites exercise a tiny fraction of our code's functionality

```
static int div(int a, int b)
{
     return a/b;
}
```

Equivalence Partitioning Motivation

- Test suites exercise a tiny fraction of our code's functionality

```
static int div(int a, int b)
{
     return a/b;
}
```

- Exhaustive testing of all values (2^{32} x 2^{32}) is not possible!

- How do we choose a subset of values to test `div`?

Equivalence Partitioning

- AKA *Equivalence Class*

- Equivalence Partitioning helps us "select a valuable fraction of the functionality most likely to be error-prone"

Equivalence Partitioning

- Divides the input data into "partitions of equivalent data" separated by boundaries (see boundary value analysis)

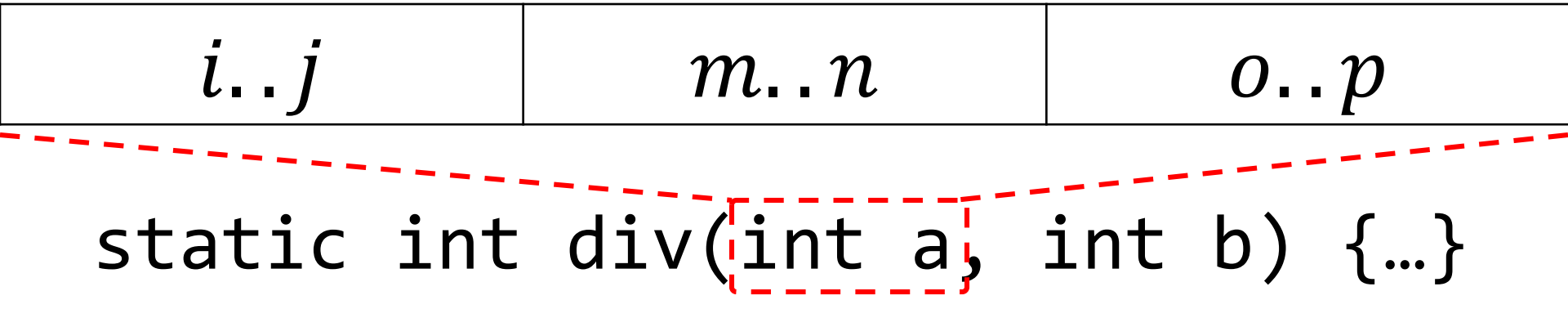
Equivalence Partitioning

- Divides the input data into "partitions of equivalent data" separated by boundaries (see boundary value analysis)

```
static int div(int a, int b) {…}
```

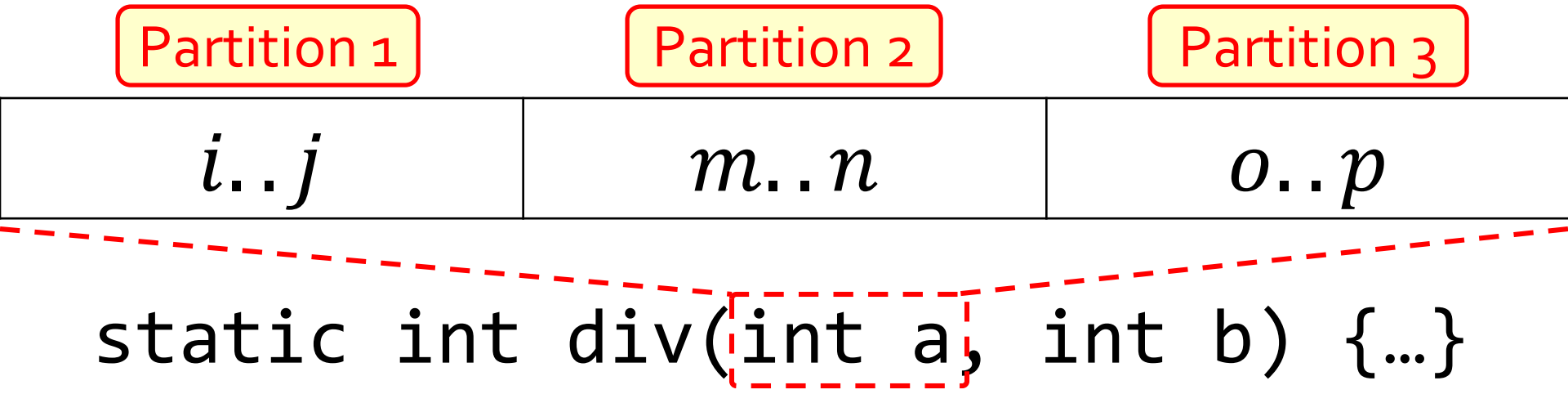
Equivalence Partitioning

- Divides the input data into "partitions of equivalent data" separated by boundaries (see boundary value analysis)



Equivalence Partitioning

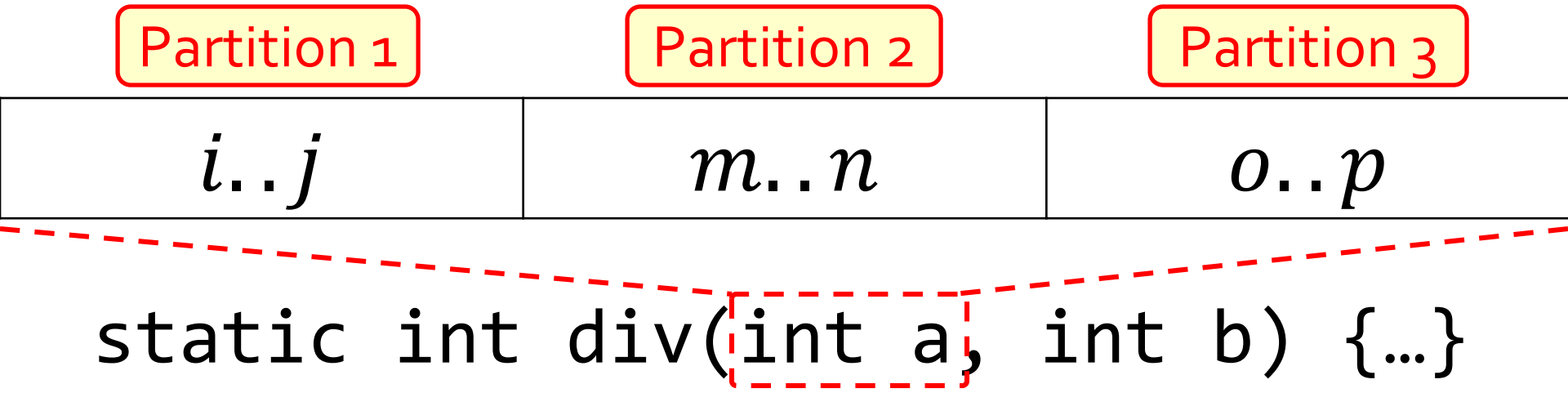
- Divides the input data into "partitions of equivalent data" separated by boundaries (see boundary value analysis)



Equivalence Partitioning

- Divides the input data into "partitions of equivalent data" separated by boundaries (see boundary value analysis)

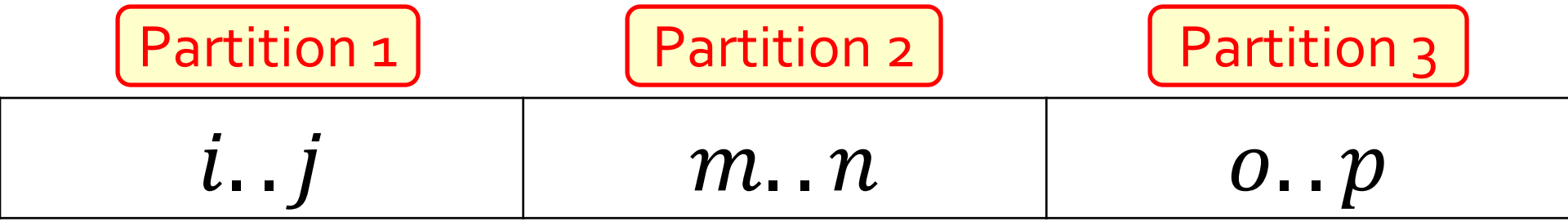
A partition represents
a region of a method's parameter input space



Equivalence Partitioning

The method behaves the same for all values within that partition/space/region

A partition represents a region of a method's parameter input space




```
static int div(int a, int b) {…}
```

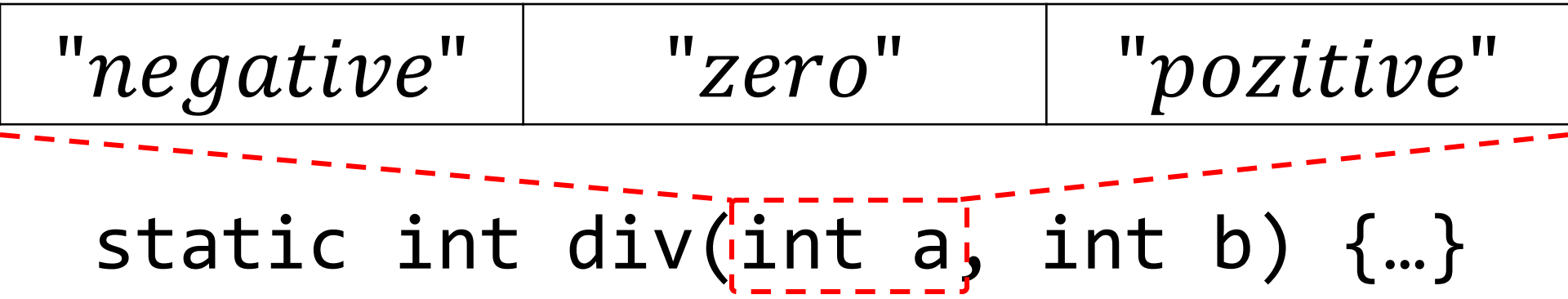
Example Equivalence Partitioning

- What are the equivalent partitions for parameter a?

```
static int div(int a, int b) {…}
```

Example Equivalence Partitioning

- What are the equivalent partitions for parameter a?



Identifying Equivalence Partitions

- How many test cases will be needed to cover the regions in the following example?

```
static int div(int a, int b)
{
    return a/b;
}
```

Identifying Equivalence Partitions

- How many test cases will be needed to cover the regions in the following example?

```
static int div(int a, int b)
{
    return a/b;
}
```

- Nine test cases:
  - a can be $\{positive, zero, negative\}$
  - b can be $\{positive, zero, negative\}$

Equivalence Partitioning

- Is a black-box technique
- Helps generate test cases

Beyond Functional Testing

- Code Coverage
- Equivalence Partitioning
- Boundary Value Analysis

Boundary Value Analysis

- Black-Box Software testing technique
- Tests are designed to include values within specified boundaries
- Input conditions are divided into groups (classes)
  - input in the same class should behave similarly in the program
  - see Equivalence Partitioning

Boundary Value Analysis Example 1

- Example from Monopoly game
- Test the feature/functionality "Go to Jail"
- Does the player have enough money to pay the $50 fine?

Boundary Value Analysis Example 1

- Example from Monopoly game
- Test the feature/functionality "Go to Jail"
- Does the player have enough money to pay the $50 fine?

$< \$50$	$\geq \$50$

Boundary Value Analysis Example 2

- Potential (integer) input range: a…b
- Test with values:

Boundary Value Analysis Example 2

- Potential (integer) input range: a...b
- Test with values:
  - a-1
  - a
  - a+1
  - some value between a and b (for equivalence partition)
  - b-1
  - b
  - b+1

Boundary Value Analysis Example 2

- Potential (integer)* input range: a…b
- Test with values:
  - a-1
  - a
  - a+1
  - some value between a and b (for equivalence partition)
  - b-1
  - b
  - b+1

*For non-integer range, test values slightly less a and slightly more than b

Boundary Value Analysis Example 3

- Months of the year expressed as integers

- Equivalent partitions are:

Boundary Value Analysis Example 3

- Months of the year expressed as integers

- Equivalent partitions are:
  - any input from the invalid partition should fail
  - any input from the valid partition should pass

$\ldots, -2, -1, 0$	$1 \ldots 12$	$13, 14, \ldots$
Invalid partition	Valid partition	Invalid partition

Other Types of Test Cases to Consider

- Can something cause division by zero?

- What if the input type is wrong
  - expecting an integer, but a float is given
  - expecting a character, but an integer is given

- What if resources (i.e., file handlers) are left open after program terminates?

These types of errors can be easily detected using static analysis tools

Other Types of Test Cases to Consider

- What if mandatory fields are not entered?
- What if the program is aborted abruptly or input or output devices are unplugged?
- What is the customer takes an "illogical" path (i.e., unexpected path) through your program?

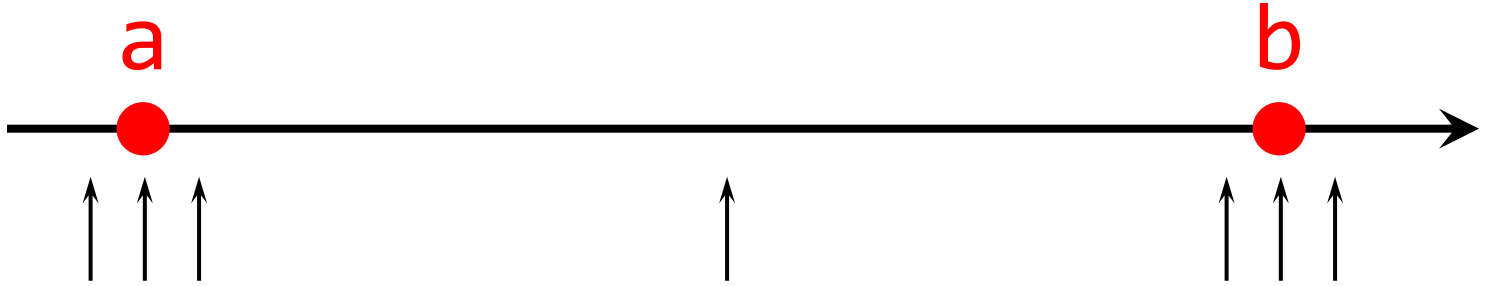
Equivalence Partitioning and Boundary Value Analysis Summary

- Defects really do tend to arise at region boundaries

- So many of our tests exercised not just one case in a region but several around the boundaries between partitions

Boundary Value Analysis Summary

- Given a range a...b we would like to test:
  - around a
  - some middle value (equivalence partition)
  - around b
- Focus is towards the boundaries



Summary

- NB: Neither *Structural Testing* (Code Coverage) and *Equivalence Partitioning* / *Value Boundary Analysis* are adequate by themselves
  - High-quality products use both

- When to stop testing?

Summary

- NB: Neither *Structural Testing* (Code Coverage) and *Equivalence Partitioning* / *Value Boundary Analysis* are adequate by themselves
  - High-quality products use both

- When to stop testing?
  - "Stop testing when fear turns into boredom"
  - Is still an open-ended question

Bibliography

- [1992Grady] Grady, Robert. Practical Software Metrics for Project Management and Process Improvement. Prentice-Hall. 1992.

- [2012Fowler]: http://martinfowler.com/bliki/TestCoverage.html

Whiteboard Only Exercise

What Type of Testing is This?

- See video of running Visual Studio Code tests at:

https://drive.google.com/file/d/1CBKD-6iQp91UiyLDW2kujFnW5tW0ZfbO/view?usp=sharing

- The tests in the videos were generated using the following command:

```
yarn smoketest
```

- Additional Resources:
  - https://github.com/Microsoft/vscode/wiki/How-to-Contribute#build-and-run-from-source
  - https://github.com/Microsoft/vscode/blob/master/test/smoke/README.md