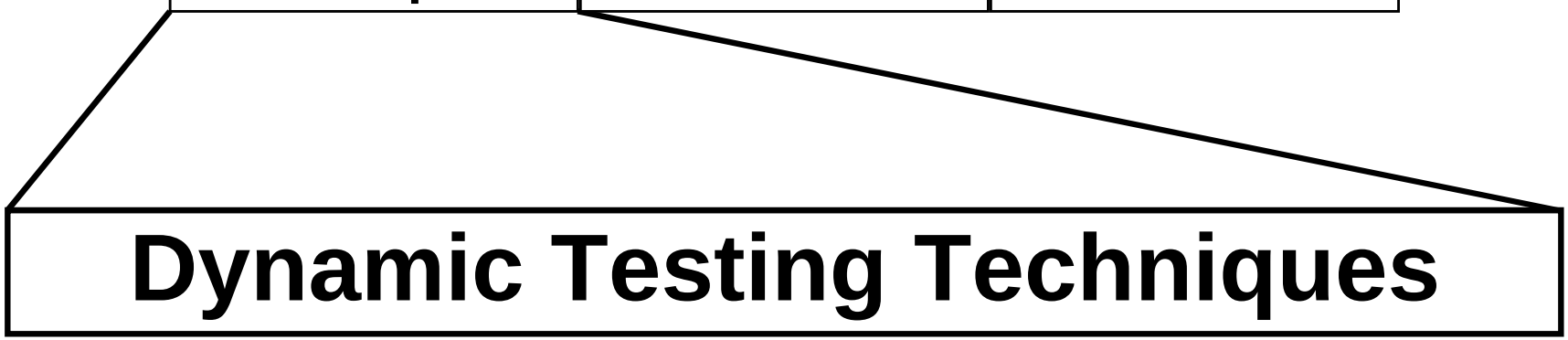
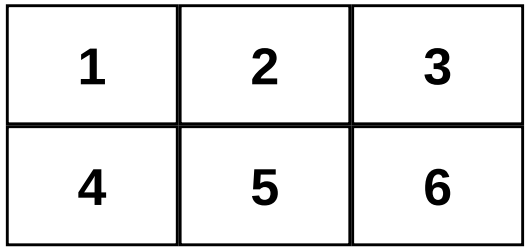
Chapter 4

Software Testing
ISTQB / ISEB Foundation Exam Practice

1 Principles	2 Lifecycle	3 Static testing
4 Dynamic test techniques	5 Management	6 Tools





Dynamic Testing Techniques

Contents

Why dynamic test techniques?

- **Exhaustive testing (use of all possible inputs and conditions) is impractical**
  - must use a subset of all possible test cases
  - must have high probability of detecting faults
- **Need thought processes that help us select test cases more intelligently**
  - test case design techniques are such thought processes

What is a testing technique?

- a procedure for selecting or designing tests
- based on a structural or functional model of the software
- successful at finding faults
- a way of deriving good test cases
- a way of objectively measuring a test effort

Testing should be rigorous, thorough and systematic

Advantages of techniques

- **Different people: similar probability find faults**
  - gain some independence of thought
- **Effective testing: find more faults**
  - focus attention on specific types of fault
  - know you're testing the right thing
- **Efficient testing: find faults with less effort**
  - avoid duplication
  - systematic techniques are measurable

Using techniques makes testing much more effective

Measurement

- **Objective assessment of thoroughness of testing (with respect to use of each technique)**
  - useful for comparison of one test effort to another
- **E.g.**

Project A	Project B
60% Equivalence partitions	40% Equivalence partitions
50% Boundaries	45% Boundaries
75% Branches	60% Branches

THE TEST DEVELOPMENT PROCESS

- The design of tests comprises three main steps:
- (1) Identify test conditions.
  - A test condition – an item or event of a component or system that could be verified by one or more test cases, e.g. a function, transaction, feature, quality attribute, or structural element.
- (2) Specify test cases.
  - A test case – a set of input values, execution preconditions, expected results and execution postconditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement.
- (3) Specify test procedures.
  - A test procedure specification – a sequence of actions for the execution of a test.

Example

- **1.2.3 The input screen shall have three fields: a title field with a drop-down selector; a surname field that can accept up to 20 alphabetic characters and the hyphen (-) character; a first name field which can accept up to 20 alphabetic characters. All alphabetic characters shall be case insensitive. All fields must be completed. The data is validated when the Enter key is pressed. If the data is valid the system moves on to the job input screen; if not, an error message is displayed.**
- **Input condition**
  - This specification enables us to define test conditions; for example, we could define a test condition for the surname field (i.e. it can accept up to 20 alphabetic characters and the hyphen (-) character) and define a set of test cases to test that field.

Test Case Design Example
we could key in the following test cases:

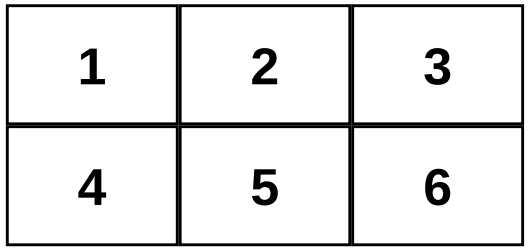
Mr Hambling Brian
Ms Samaroo Angelina
Ms Simmonite Compo
Mr Hyde-White Wilfred

We should also test some invalid inputs

Mr Thompson1 Geoff
Mr “Morgan” Peter
Mr Williams ‘Pete’

The test procedure would need to add some details along the following lines:

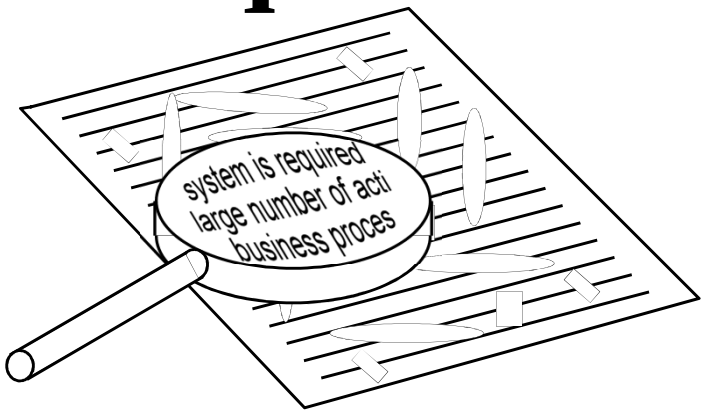
- (1) Select the <Name or Personal Details> option from the main menu.
- (2) Select the ‘input’ option from the <Name or Personal Details> menu.
- (3) Select ‘Mr’ from the ‘Title’ drop-down menu.
- (4) Check that the cursor moves to the ‘surname’ field.
- (5) Type in ‘Hambling’ and press the tab key once; check that the cursor moves to the ‘first name’ field.
- (6) Type in ‘Brian’ and press the Enter key.
- (7) Check that the Job Input screen is displayed.



Dynamic Testing Techniques

Contents

Three types of systematic technique

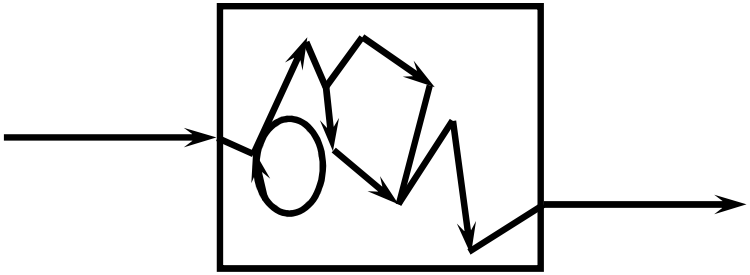


Static (non-execution)

- examination of documentation, source code listings, etc.

Functional (Black Box)

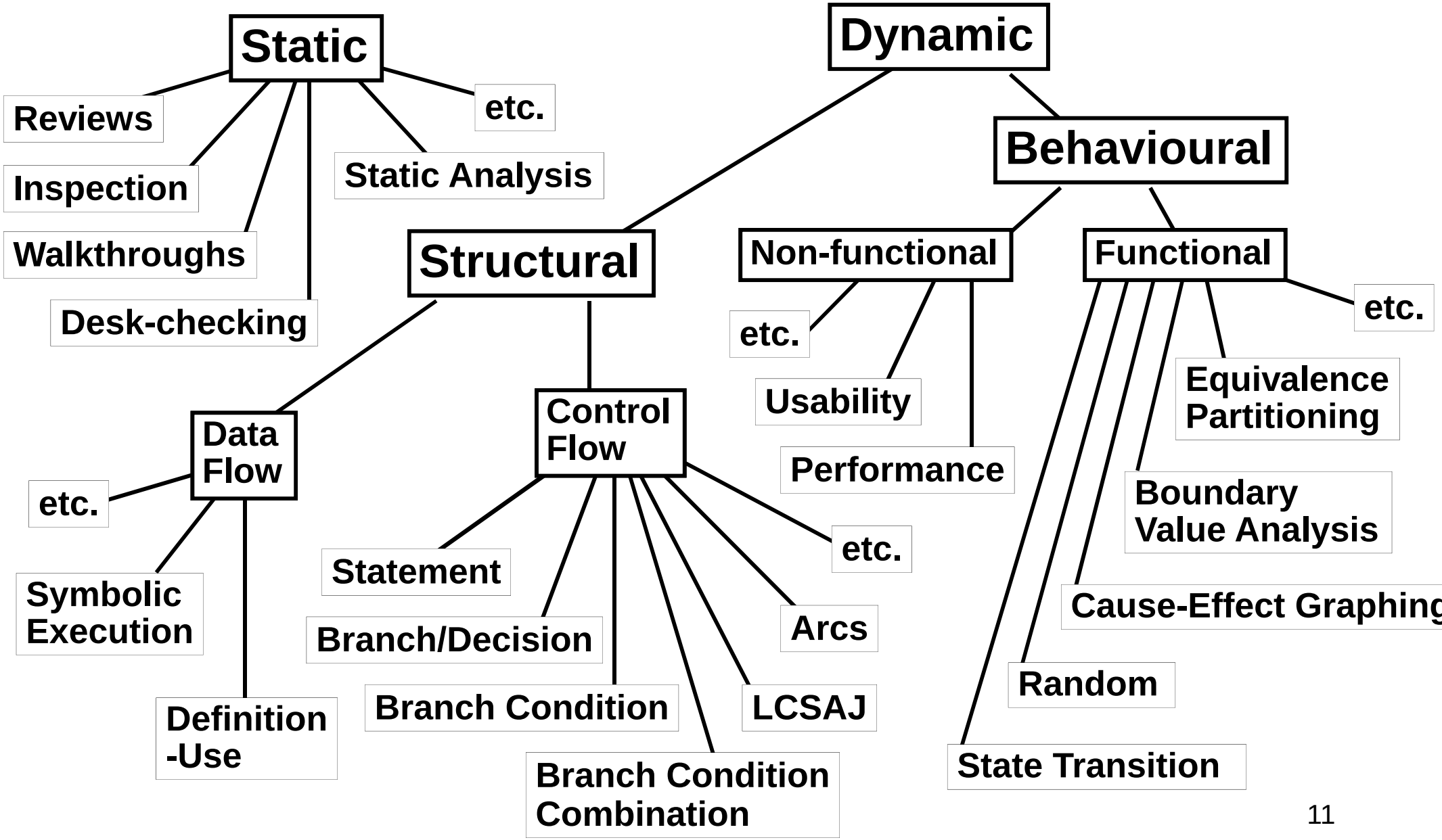
- based on behaviour / functionality of software



Structural (White Box)

- based on structure of software

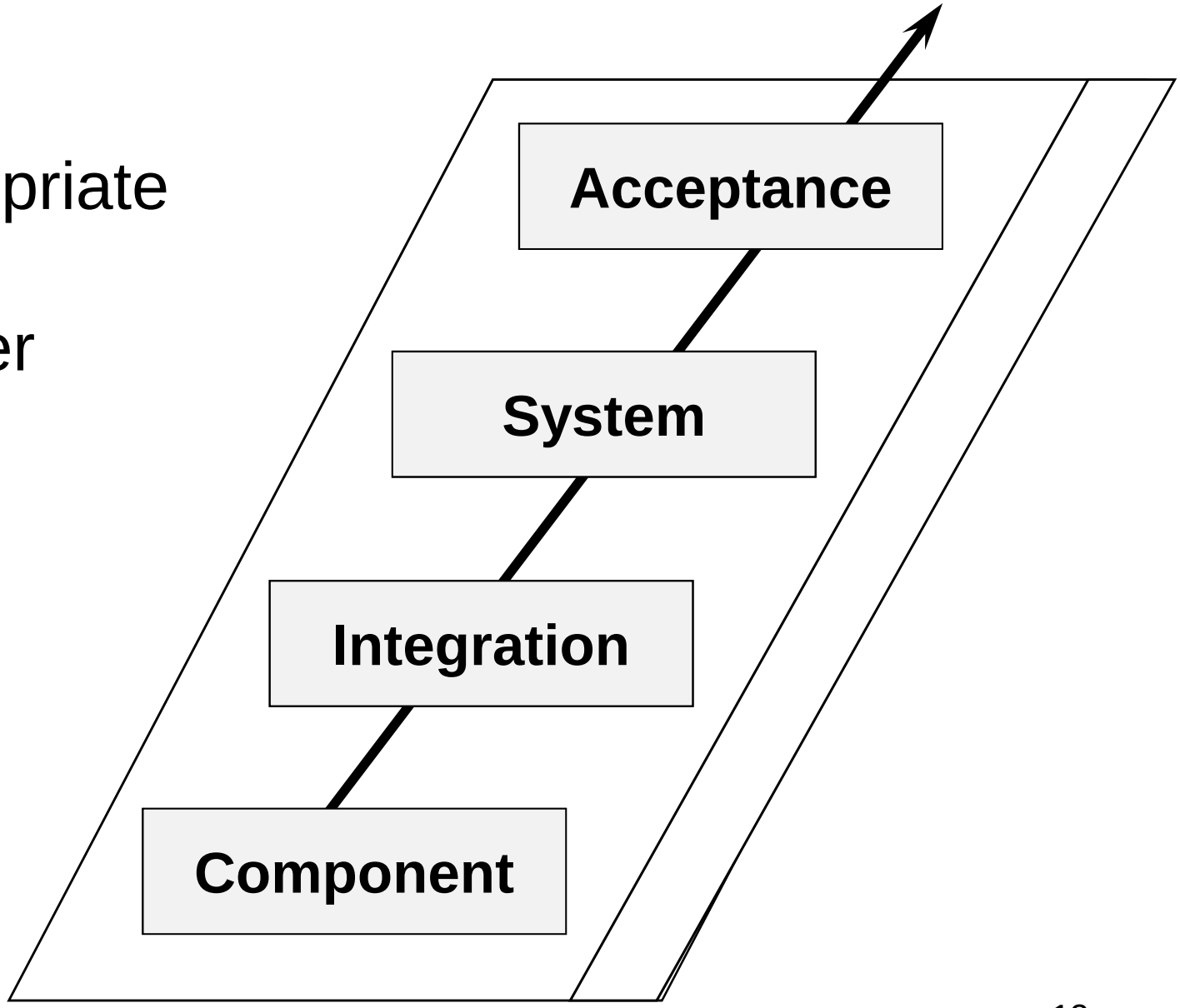
Some test techniques

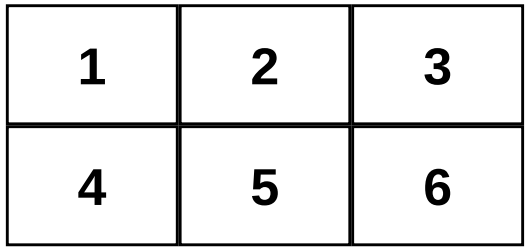


Black box versus white box?

Black box appropriate at all levels but dominates higher levels of testing

White box used predominately at lower levels to compliment black box





Dynamic Testing Techniques

Contents

Black Box test design and measurement techniques

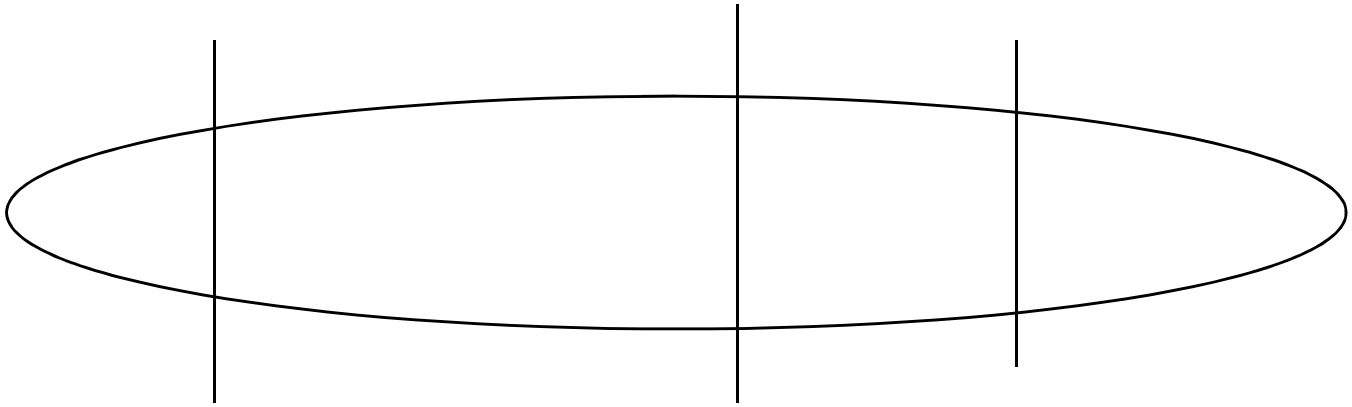
- **Techniques defined in BS 7925-2**
  - Equivalence partitioning ✓
  - Boundary value analysis ✓
  - State transition testing ✓
  - Cause-effect graphing ✓
  - Syntax testing ✗
  - Random testing ✗
- **Also defines how to specify other techniques**

Also a measurement technique?

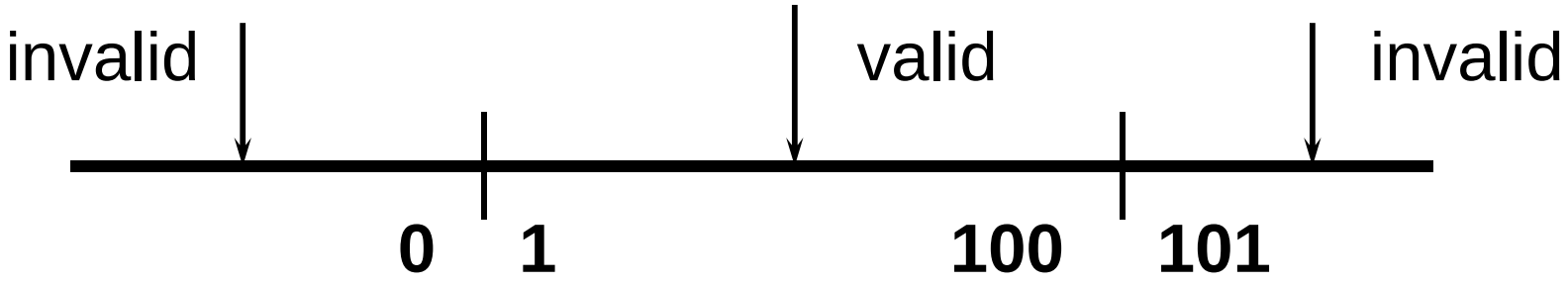
✓ **= Yes**

✗ **= No**

Equivalence partitioning (EP)



- divide (partition) the inputs, outputs, etc. into areas which are the same (equivalent)
- assumption: if one value works, all will work
- one from each partition better than all from one



EXAMPLE EQUIVALENCE PARTITIONS

- Valid input: integers in the range 100 to 999.
  - Valid partition: 100 to 999 inclusive.
  - Non-valid partitions: less than 100, more than 999, real (decimal) numbers and non-numeric characters.
- Valid input: names with up to 20 alphabetic characters.
  - Valid partition: strings of up to 20 alphabetic characters.
  - Non-valid partitions: strings of more than 20 alphabetic characters, strings containing non-alphabetic characters.

- Exercise 4.1
- Suppose you have a bank account that offers variable interest rates:
  - 0.5 percent for the first £1,000 credit;
  - 1 percent for the next £1,000;
  - 1.5 percent for the rest.
- If you wanted to check that the bank was handling your account correctly what valid input partitions might you use?

- **PARTITIONS – EXAMPLE 4.1**
- A mail-order company charges £2.95 postage for deliveries if the package weighs less than 2 kg, £3.95 if the package weighs 2 kg or more but less than 5 kg, and £5 for packages weighing 5 kg or more.
- Generate a set of valid test cases using equivalence partitioning.

- The valid input partitions are: under 2 kg; 2 kg or over but less than 5 kg; and 5 kg or over.

- Input values could be 1 kg, 3.5 kg, 7.5 kg. These would produce expected results of £2.95, £3.95 and £5 respectively.
- In this case there are no non-valid inputs (unless the scales fail).

- Exercise 4.2
- A mail-order company selling flower seeds charges £3.95 for postage and packing on all orders up to £20 value and £4.95 for orders above £20 value and up to £40 value. For orders above £40 value there is no charge for postage and packing.

- If you were using equivalence partitioning to prepare test cases for the postage and packing charges what valid partitions would you define?
- What about non-valid partitions?
- The answer can be found at the end of the chapter

- Example
- A savings account in a bank earns a different rate of interest depending on the balance in the account. In order to test the software that calculates the interest due, we can identify the ranges of balance values that earn the different rates of interest. For example, if a balance in the range $0 up to $100 has a 3% interest rate, a balance over $100 and up to $1000 has a 5% interest rate, and balances of $1000 and over have a 7% interest rate, we would initially identify three valid equivalence partitions and one invalid partition as shown below.

Invalid partition	Valid (for 3% interest)		Valid (for 5%)		Valid (for 7%)
-$0.01	$0.00	$100.00	$100.01	$999.99	$1000.00

Example-Triangles

- **“A program reads three integer values. The three values are interpreted as representing the lengths of the sides of a triangle. The program prints a message that states whether the triangle is scalene, isosceles , or equilateral.”**

- **Write a set of test cases to test this program.**

- **You can skip this example**

Answer

1. valid scalene triangle ?
2. valid equilateral triangle ?
3. valid isosceles triangle ?
4. 3 permutations of previous ?
5. side = 0 ?
6. negative side ?
7. one side is sum of others ?
8. 3 permutations of previous ?
9. one side larger than sum of others ?
10. 3 permutations of previous ?
11. all sides = 0 ?
12. non-integer input ?
13. wrong number of values ?
14. for each test case: is expected output specified ?
15. check behaviour after output was produced ?

Equivalence Classes for Variables: Range

Eq. Classes	Example	
	Constraints	Classes
One class with values inside the range and two with values outside the range.	speed $\in$[60..90]	{50}, {75}, {92}
	area: float area$\geq$0.0	{{-1.0}, {15.52}}
	age: int	{{-1}, {56}, {132}}

Equivalence Classes for Variables: Strings

Eq. Classes	Example	
At least one containing all legal strings and one all illegal strings based on any constraints.	firstname: string	{{ε}, {Sue}, {Loooong Name}}

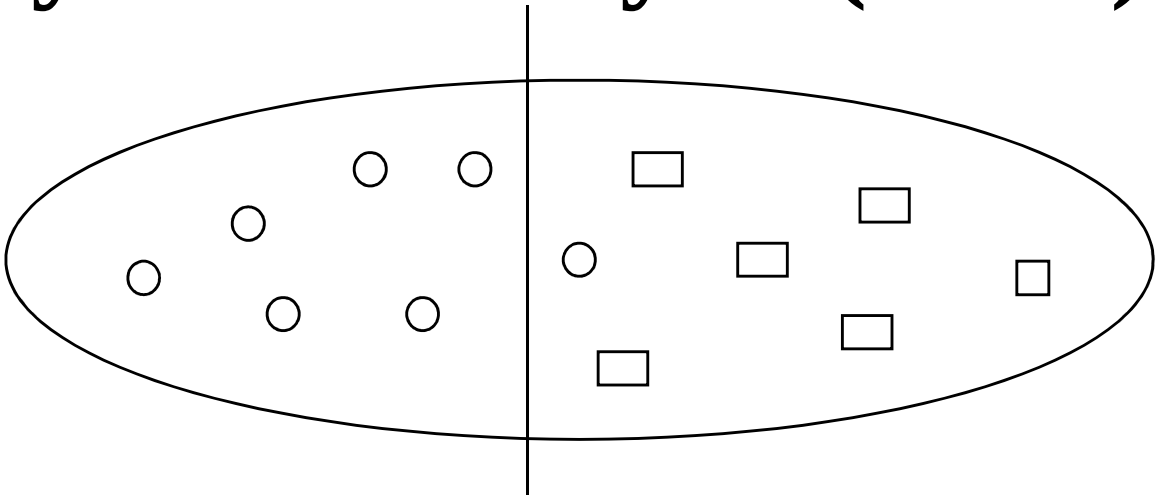
Equivalence Classes for Variables: Enumeration

Eq. Classes	Example	
	Constraints	Classes
Each value in a separate class	autocolor:{red, blue, green}	{{red}, {blue}, {green}}
	up:boolean	{{true}, {false}}

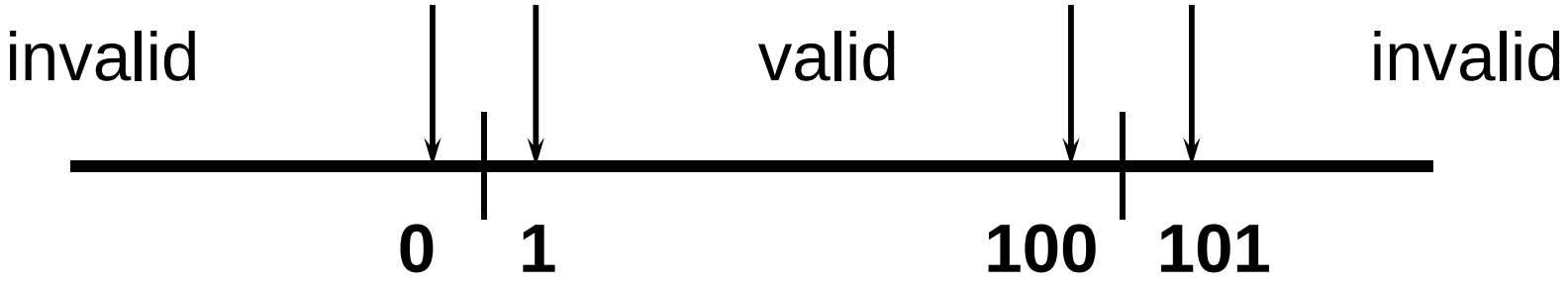
Equivalence Classes for Variables: Arrays

Eq. Classes	Example	
	Constraints	Classes
One class containing all legal arrays, one containing the empty array, and one containing a larger than expected array.	int [ ] aName = new int[3];	{[ ]}, {[-10, 20]}, {[-9, 0, 12, 15]}

Boundary value analysis (BVA)



- faults tend to lurk near boundaries
- good place to look for faults
- test values on both sides of boundaries



BVA guidelines from another book

- 1. If an input condition specifies a range of values, write test cases for the ends of the range, and invalid-input test cases for situations just beyond the ends.
  - For instance, if the valid domain of an input value is 1.0, write test cases for the situations 1.0,-1.001, and 1.001.
- 2. If an input condition specifies a number of values, write test cases for the minimum and maximum number of values and one beneath and beyond these values.
  - For instance, if an input file can contain 1–255 records, write test cases for 0, 1, 255, and 256 records.

BVA guidelines from another book

- 3. Use guideline 1 for each OUTPUT CONDITION.
  - For instance, if a program computes the monthly FICA deduction and if the minimum is $0.00 and the maximum is $1,165.25, write test cases that cause
    - $0.00 and $1,165.25 to be deducted.
    - Also, see if it is possible to invent test cases that might cause a negative deduction or a deduction of more than $1,165.25.
  - Note that it is important to examine the boundaries of the result space because it is not always the case that the boundaries of the input domains represent the same set of circumstances as the boundaries of the output ranges (e.g., consider a sine subroutine).
  - Also, it is not always possible to generate a result outside of the output range, but it is worth considering the possibility, nonetheless.

BVA guidelines from another book

- 4. Use guideline 2 for each output condition.
  - If an information-retrieval system displays the most relevant abstracts based on an input request, but never more than four abstracts, write test cases such that the program displays zero, one, and four abstracts, and write a test case that might cause the program to erroneously display five abstracts.
- 5. If the input or output of a program is an ordered set (a sequential file, for example, or a linear list or a table), focus attention on the first and last elements of the set.
- 6. In addition, use your ingenuity to search for other boundary conditions.

BOUNDARY VALUES – EXAMPLE 4.2

The boiling point of water – the boundary is at 100 degrees Celsius, so for the 3 Value Boundary approach the boundary values will be

- 99 degrees, 100 degrees, 101 degrees

unless you have a very accurate digital thermometer, in which case they could be

- 99.9 degrees, 100.0 degrees, 100.1 degrees.

For the 2 value approach the corresponding values would be 100 and 101.

Exam pass – if an exam has a pass boundary at 40 percent, merit at 60 percent and distinction at 80 percent the 3 value boundaries would be

- 39, 40, 41 for pass,
- 59, 60, 61 for merit,
- 79, 80, 81 for distinction.

It is unlikely that marks would be recorded at any greater precision than whole numbers.

The 2 value equivalents would be 39 and 40, 59 and 60, and 79 and 80 respectively.

BOUNDARY VALUES – EXAMPLE 4.2

- The boiling point of water – the boundary is at 100 degrees Celsius, so for the 3 Value Boundary approach the boundary values will be 99 degrees, 100 degrees, 101 degrees – unless you have a very accurate digital thermometer, in which case they could be 99.9 degrees, 100.0 degrees, 100.1 degrees. For the 2 value approach the corresponding values would be 100 and 101.
- Exam pass – if an exam has a pass boundary at 40 per cent, merit at 60 per cent and distinction at 80 per cent the 3 value boundaries would be 39, 40, 41 for pass, 59, 60, 61 for merit, 79, 80, 81 for distinction. It is unlikely that marks would be recorded at any greater precision than whole numbers. The 2 value equivalents would be 39 and 40, 59 and 60, and 79 and 80 respectively.

Assuming that an item code must be in the range 99..999 and quantity in the range 1..100

Equivalence classes for code

E1: Values less than 99

E2: Values in the range

E3: Values greater than 999

Equivalence classes for qty

E4: Values less than 1

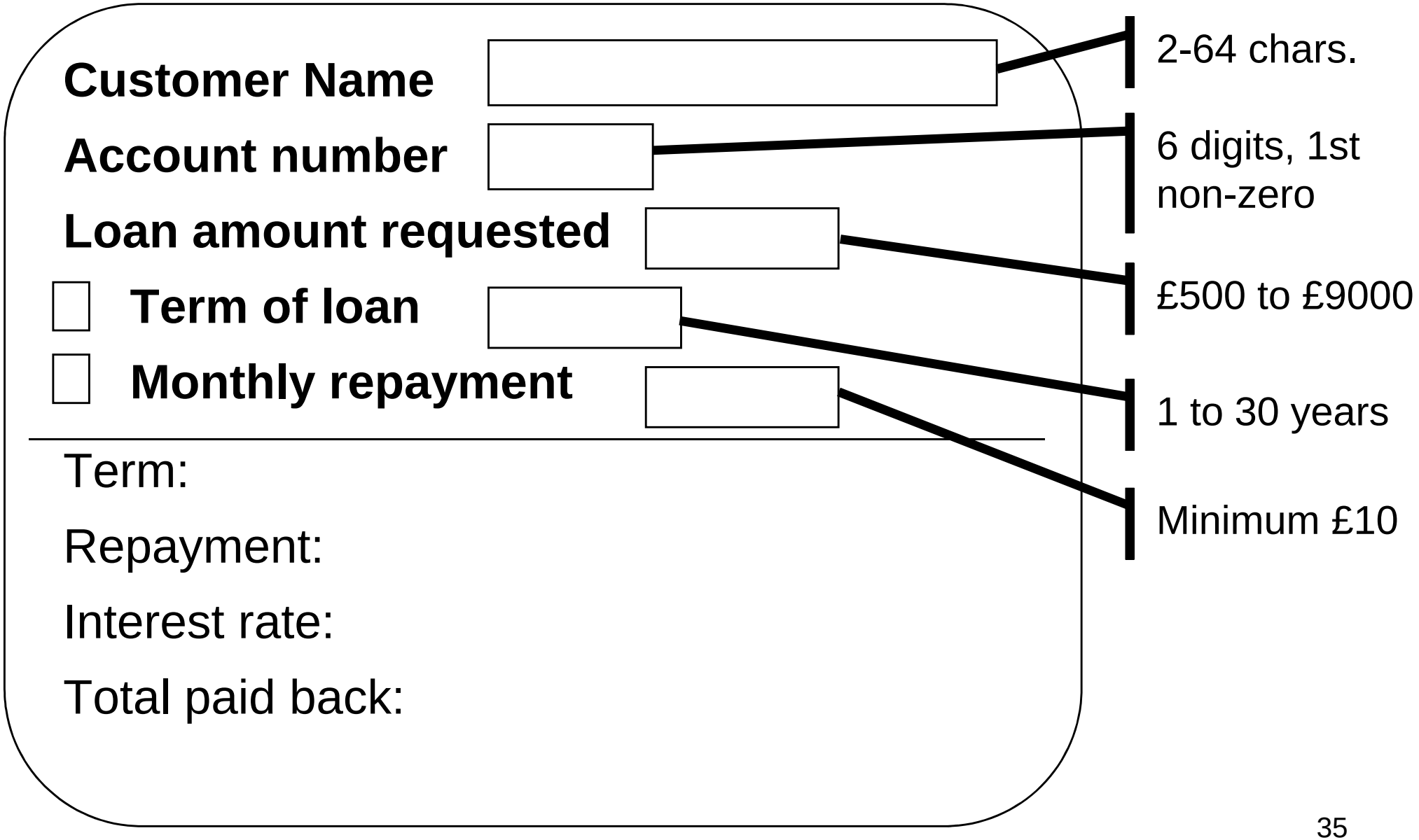
E5: Values in the range

E6: Values greater than 100

TABLE 4.1 Equivalence partitions and boundaries

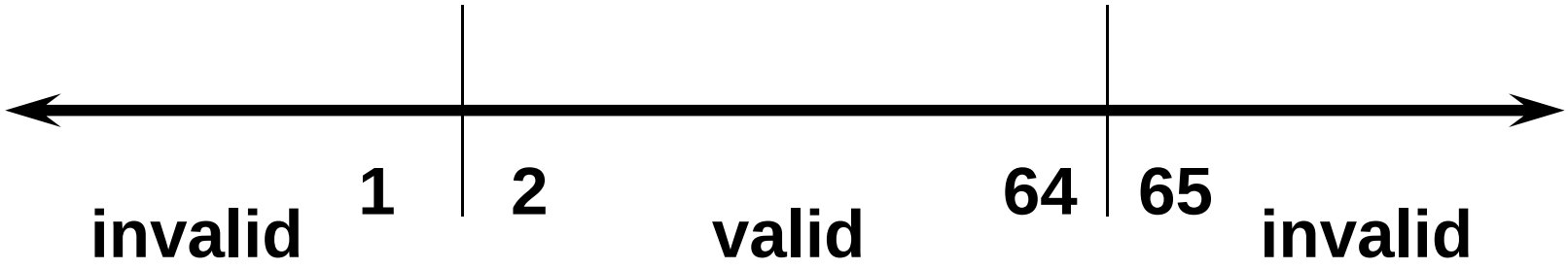
Test conditions	Valid partitions	Invalid partitions	Valid boundaries	Invalid boundaries
Balance in aU$_o$Unt	$0.00 $100.00	< $0.00	$0.00	-$0.01
	$100.01-$999.99	>$Max	$100.00	$Max+0.01
	$1000.00- $Max	non-integer (if balance is an input field)	$100.01	
			$999.99	
			$1000.00	
			$Max	
Interest rates	3%	Any other value	Not applicable	Not applicable
	5%	Non-integer		
	7%	No interest calculated		

Example: Loan application

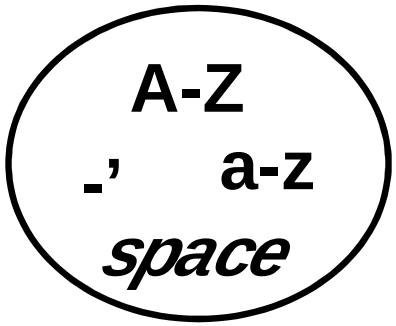


Customer name

Number of characters:



Valid characters:



Any other

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Customer name	2 to 64 chars	< 2 chars	2 chars	1 chars
	valid chars	> 64 chars	64 chars	65 chars
		invalid chars		0 chars

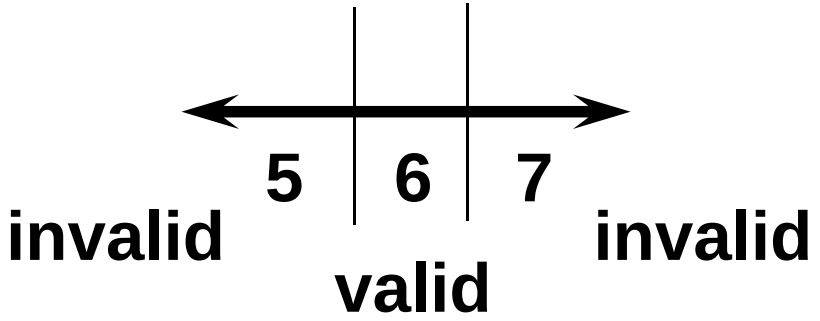
Account number

first character:

valid: non-zero

invalid: zero

number of digits:



Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Account number	6 digits	< 6 digits	100000	5 digits
	1st non-zero	> 6 digits	999999	7 digits
		1st digit = 0		0 digits
		non-digit		

Loan amount

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Loan amount	500 - 9000	< 500	500	499
		>9000	9000	9001
		0		
		non-numeric		
		null		

Condition template

Conditions	Valid Partitions	Tag	Invalid Partitions	Tag	Valid Boundaries	Tag	Invalid Boundaries	Tag
Customer name	2 - 64 chars	V1	< 2 chars	X1	2 chars	B1	1 char	D1
	valid chars	V2	> 64 chars	X2	64 chars	B2	65 chars	D2
			invalid char	X3			0 chars	D3
Account number	6 digits	V3	< 6 digits	X4	100000	B3	5 digits	D4
	1st non-zero	V4	> 6 digits	X5	999999	B4	7 digits	D5
			1st digit = 0	X6			0 digits	D6
			non-digit	X7				
Loan amount	500 - 9000	V5	< 500	X8	500	B5	499	D7
			>9000	X9	9000	B6	9001	D8
			0	X10				
			non-integer	X11				
			null	X12				

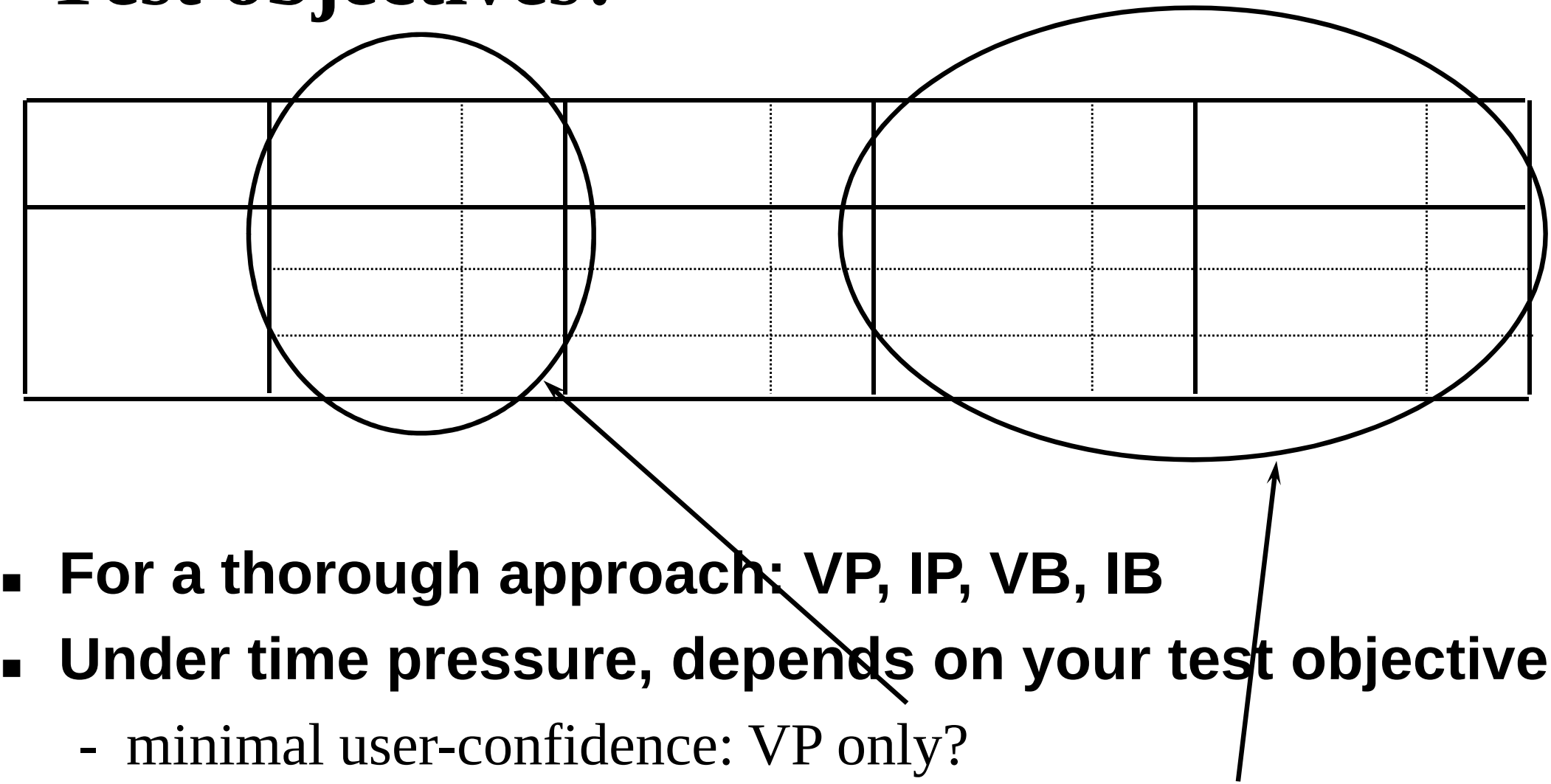
Design test cases

Test Case	Description	Expected Outcome	New Tags Covered
1	Name: John Smith Acc no: 123456 Loan: 2500 Term: 3 years	Term: 3 years Repayment: 79.86 Interest rate: 10% Total paid: 2874.96	V1, V2, V3, V4, V5 .....
2	Name: AB Acc no: 100000 Loan: 500 Term: 1 year	Term: 1 year Repayment: 44.80 Interest rate: 7.5% Total paid: 537.60	B1, B3, B5, ...

Why do both EP and BVA?

- **If you do boundaries only, you have covered all the partitions as well**
  - technically correct and may be OK if everything works correctly!
  - if the test fails, is the whole partition wrong, or is a boundary in the wrong place - have to test mid-partition anyway
  - testing only extremes may not give confidence for typical use scenarios (especially for users)
  - boundaries may be harder (more costly) to set up

Test objectives?



- **For a thorough approach: VP, IP, VB, IB**
- **Under time pressure, depends on your test objective**
  - minimal user-confidence: VP only?
  - maximum fault finding: VB first (plus IB?)

Decision tables

- **explore combinations of inputs, situations or events,**
- **it is very easy to overlook specific combinations of input**
- **start by expressing the input conditions of interest so that they are either TRUE or FALSE**
  - record found
  - file exists
  - code valid
  - policy expired
  - account in credit
  - due date > current date

DECISION TABLE STRUCTURE

	Business rule 1	Business rule 2	Business rule 3
Condition 1	T	F	T
Condition 2	T	T	T
Condition 3	T	–	F
Action 1	Y	N	Y
Action 2	N	Y	Y

Example: student access

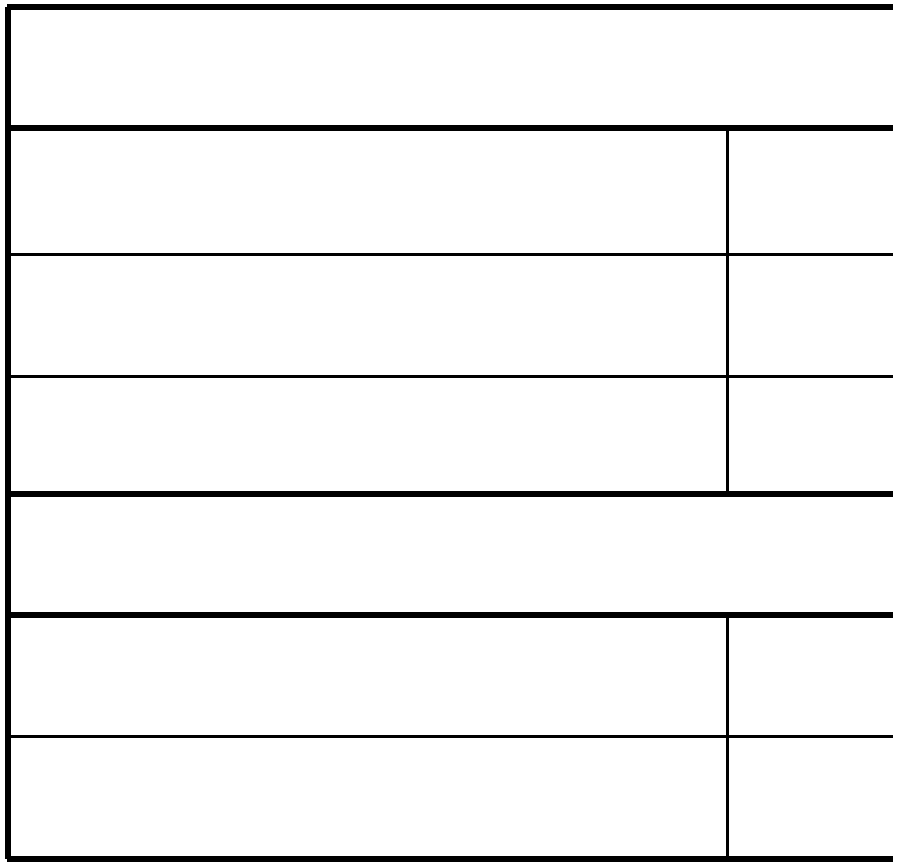
A university computer system allows students an allocation of disc space depending on their projects.

If they have used all their allotted space, they are only allowed restricted access, i.e. to delete files, not to create them. This is assuming they have logged on with a valid username and password.

What are the input and output conditions?

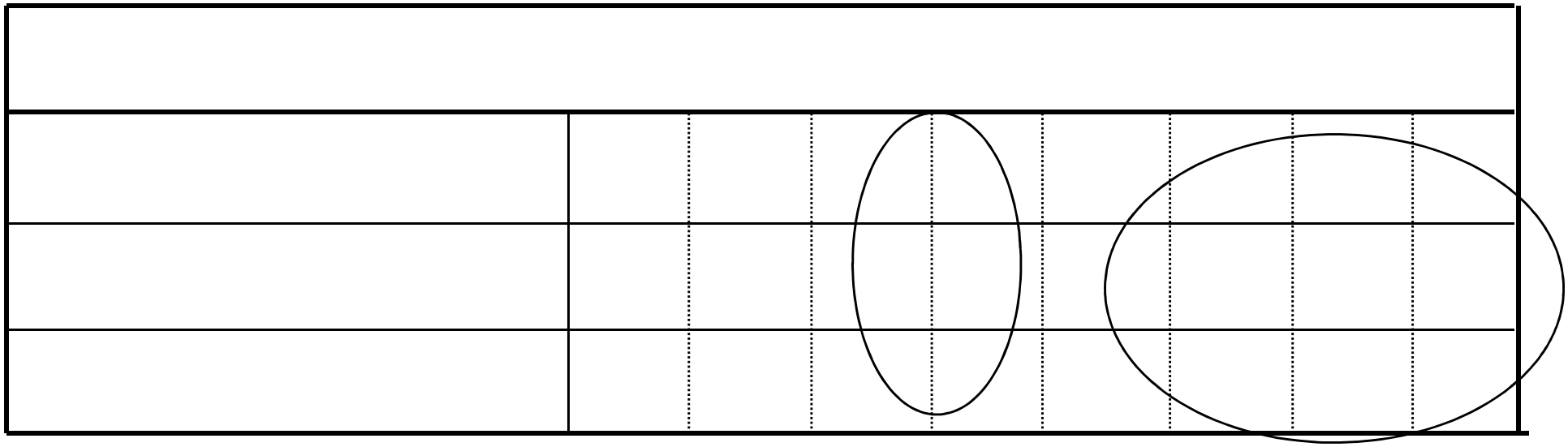
List the input and output conditions

- list the ‘input conditions’ in the first column of the table
- list the ‘output conditions’ under the output conditions



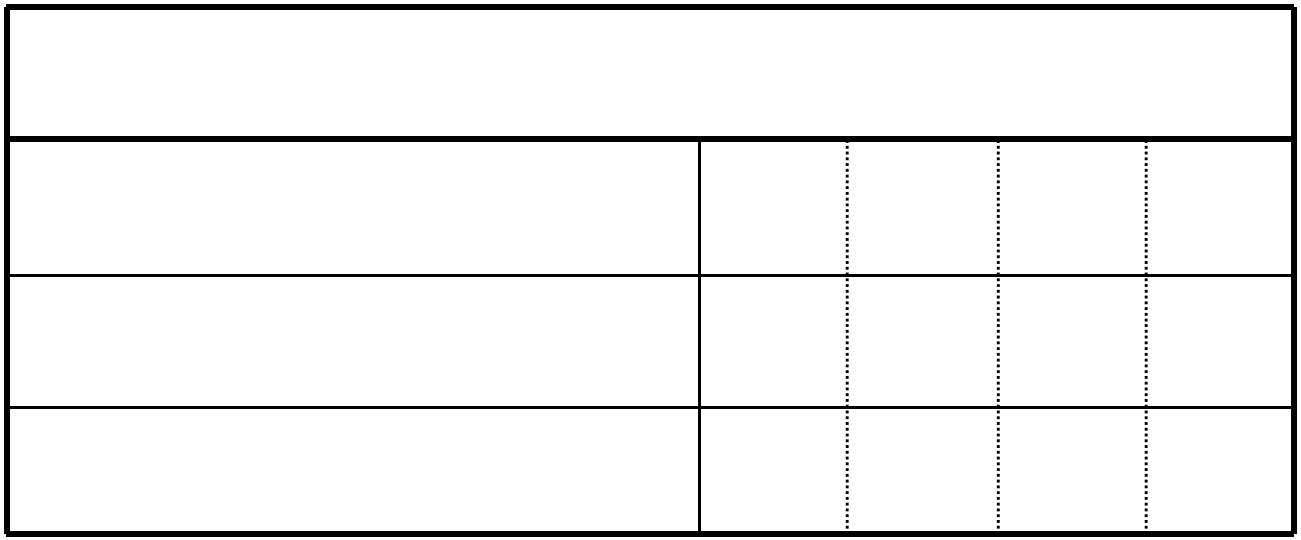
Determine input combinations

- add columns to the table for each unique combination of input conditions.
- each entry in the table may be either ‘T’ for true, ‘F’ for false.



Rationalise input combinations

- some combinations may be impossible or not of interest
- some combinations may be ‘equivalent’
- use a hyphen to denote “don’t care”

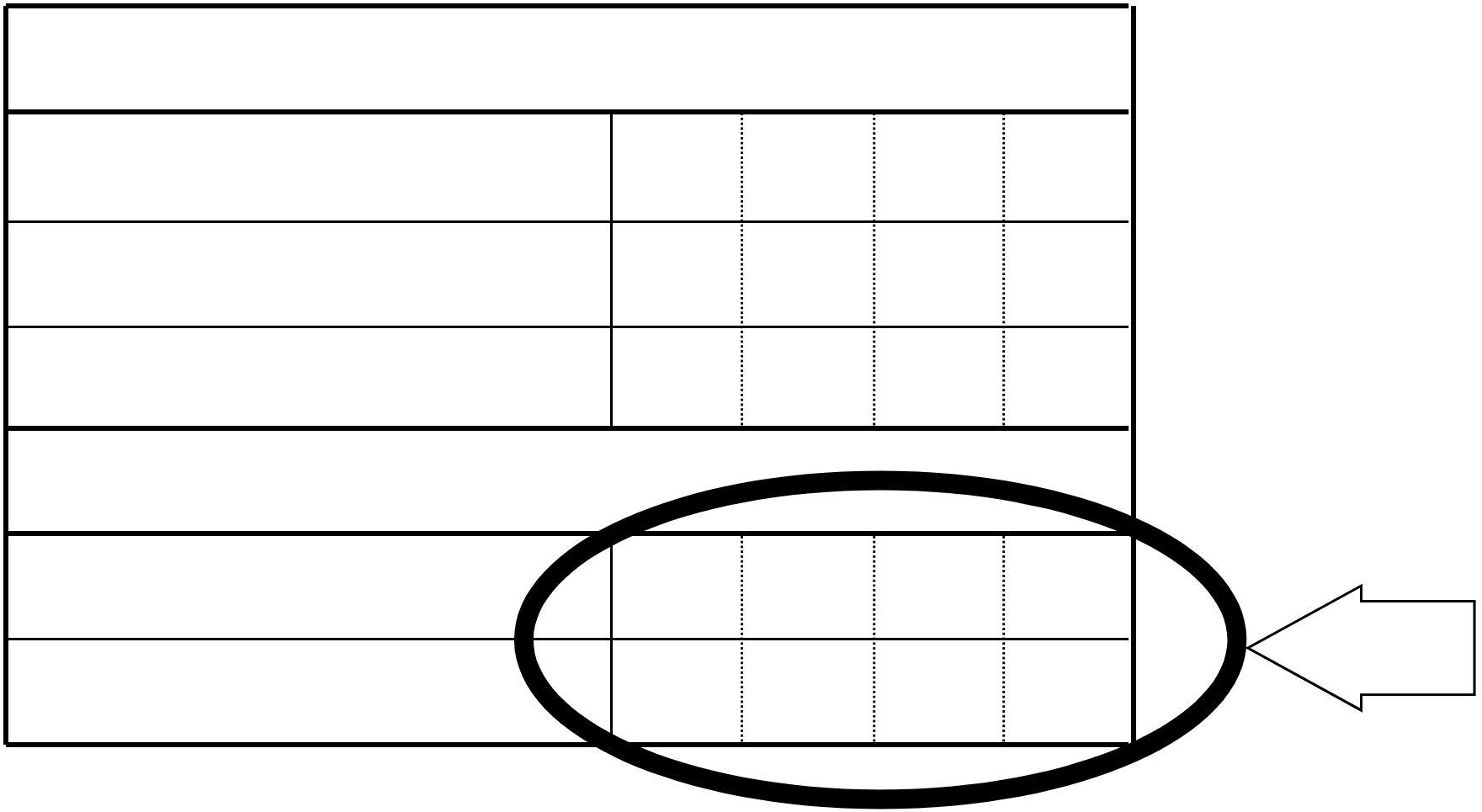


Input Conditions								
Valid username	T	T	T	T	F	F	F	F
Valid password	T	T	F	F	T	T	F	F
Account in credit	T	F	T	F	T	F	T	F
login accepted	T	T	F	F	F	F	F	F
restricted access	F	T	F	F	F	F	F	F

Input Conditions								
Valid username	T	T	T	T	F	F	F	F
Valid password	T	T	F	F	-	-	-	-
Account in credit	T	F	-	-	-	-	-	-
login accepted	T	T	-	-	-	-	-	-
restricted access	F	T	-	-	-	-	-	-

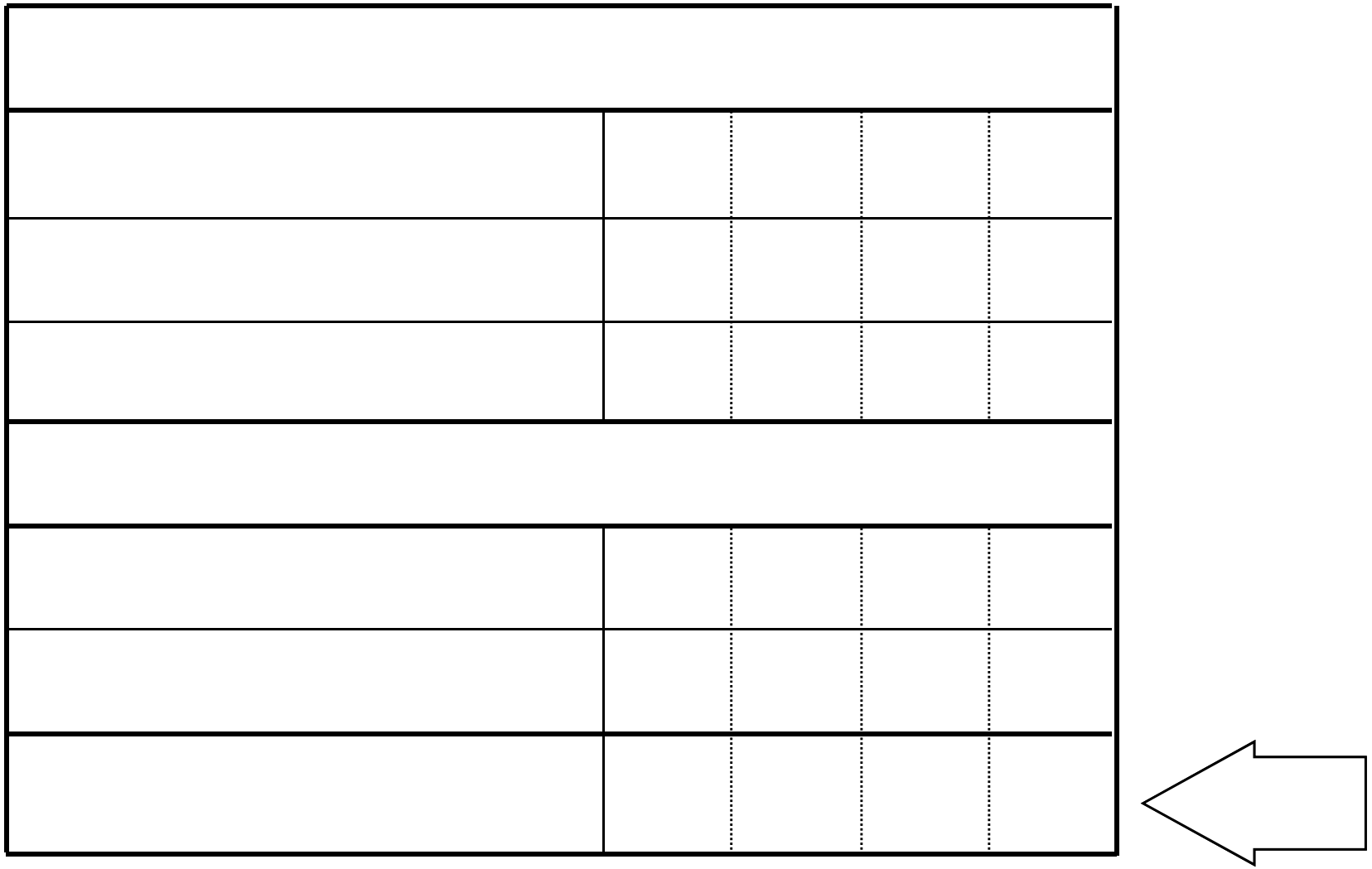
Complete the table

- **determine the expected output conditions for each combination of input conditions**



Determine test case groups

- each column is at least one test case

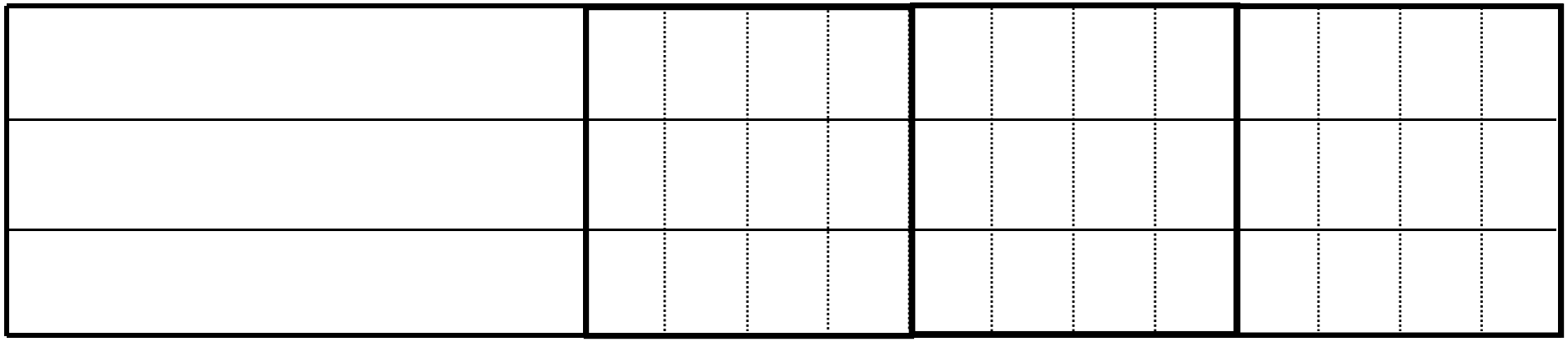


Design test cases

- usually one test case for each column but can be none or several

Extending decision tables

- **Entries can be more than just ‘true’ or ‘false’**
  - completing table needs to be done carefully
  - rationalising becomes more important
- **E.g.**



- **A marketing company wishes to construct a decision table to decide how to treat clients according to three characteristics:**
  - Gender,
  - City Dweller, and
  - age group: A (under 30), B (between 30 and 60), C (over 60).
- **The company has four products (W, X, Y and Z) to test market.**
  - Product W will appeal to male city dwellers.
  - Product X will appeal to young males.
  - Product Y will appeal to Female middle aged shoppers who do not live in cities.
  - Product Z will appeal to all but older males.

- **Identify Conditions & Values**
- **The three data attributes tested by the conditions in this problem are**
  - gender, with values M and F;
  - city dweller, with value Y and N; and
  - age group, with values A, B, and C
- **as stated in the problem.**

- **Compute Maximum Number of Rules**

- **The maximum number of rules is 2 x 2 x 3 = 12**

- **Identify Possible Actions**
- **• The four actions are:**
  - market product W,
  - market product X,
  - market product Y,
  - market product Z.

Enter All Possible Conditions

RULES												
	1	2	3	4	5	6	7	8	9	10	11	12
Sex	m	f	m	f	m	f	m	f	m	f	m	f
City	y	y	n	n	y	y	n	n	y	Y	n	n
Age	a	a	a	a	b	b	b	b	c	c	c	c

– gender, with values M and F;
– city dweller, with value Y and N; and
– age group, with values A, B, and C

Define Actions for each Rule

Actions												
Market	1	2	3	4	5	6	7	8	9	10	11	12
W	X				X				X			
X	X		X									
Y								X				
Z	X	X	X	X	X	X	X	X		X		X

- Product W will appeal to male city dwellers.
- Product X will appeal to young males.
- Product Y will appeal to Female middle aged shoppers who do not live in cities.
- Product Z will appeal to all but older males.

RULES												
	1	2	3	4	5	6	7	8	9	10	11	12
Sex	m	f	m	f	m	f	m	f	m	f	m	f
City	y	y	n	n	y	y	n	n	y	Y	n	n
Age	a	a	a	a	b	b	b	b	c	c	c	c
Actions												
Market	1	2	3	4	5	6	7	8	9	10	11	12
W	X				X				X			
X	X		X									
Y								X				
Z	X	X	X	X	X	X	X	X		X		X

- Product W will appeal to male city dwellers.
- Product X will appeal to young males.
- Product Y will appeal to Female middle aged shoppers who do not live in cities.
- Product Z will appeal to all but older males.

RULES												
	1	2	3	4	5	6	7	8	9	10	11	12
Sex	m	f	m	f	m	f	m	f	m	f	m	f
City	y	y	n	n	y	y	n	n	y	Y	n	n
Age	a	**a**	a	a	b	**b**	b	b	c	c	c	c
Actions												
Market	1	2	3	4	5	6	7	8	9	10	11	12
W	X				X				X			
X	X		X									
Y								X				
Z	X	X	X	X	X	X	X	X		X		X

- two of the three condition values (gender and city dweller) identical and
- all three of the values of the non-identical value (age) are covered,
- so they can be condensed into a single column 2

The revised table is as follows:

RULES										
	1	2	3	4	5	6	7	8	9	10
Sex	M	F	M	F	M	M	F	M	M	F
City	Y	Y	N	N	Y	N	N	Y	N	N
Age	A	-	A	A	B	B	B	C	C	C
Actions										
Market	1	2	3	4	5	6	7	8	9	10
W	X				X			X		
X	X		X							
Y							X			
Z	X	X	X	X	X	X	X			X

Simplify the Table

- There appear to be no impossible rules.
- Note that rules 2, 4, 6, 10, 12 have the same action pattern.
- Rules 2, 6 and 10 have
  - two of the three condition values (gender and city dweller) identical and
  - all three of the values of the non-identical value (age) are covered,
- so they can be condensed into a single column 2.
- The rules 4 and 12 have identical action pattern, but they cannot be combined because the indifferent attribute "Age" does not have all its values covered in these two columns.
- Age group B is missing.

- In this example we are going to test the following requirements.

- The system shall only calculate discounts for members.
- The system shall calculate a discount of 5% if the value of the purchase is less than or equal to 100. Otherwise the discount is 10%.
- The system shall write the discount % on the invoice.
- The system must write in the invoices to non members that membership gives a discount.

Conditions	**R1**	**R2**	**R3**	**R4**
Purchaser is member	T	T	F	F
Value <= € 100	T	F	T	F
Actions				
5% discount calculated	T	F	F	F
10% discount calculated	F	T	F	F
Member message on invoice	F	F	T	T
Discount % on invoice	T	T	F	F

Conditions	**R1**	**R2**	**R3**
Purchaser is member	T	T	F
Value <= € 100	T	F	-
Actions			
5% discount calculated	T	F	F
10% discount calculated	F	T	F
Member message on invoice	F	F	T
Discount % on invoice	T	T	F

Example

- Take an example of transferring money online to an account which is already added and approved.
- Here the conditions to transfer money are
  - ACCOUNT ALREADY APPROVED, OTP (One Time Password) MATCHED, SUFFICIENT MONEY IN THE ACCOUNT.
- And the actions performed are
  - TRANSFER MONEY, SHOW A MESSAGE AS INSUFFICIENT AMOUNT, BLOCK THE TRANSACTION INCASE OF SUSPICIOUS TRANSACTION.

DECISION TABLE

ID	CONDITIONS/ACTIONS	TEST CASE 1	TEST CASE 2	TEST CASE 3	TEST CASE 4	TEST CASE 5
Condition 1	Account Already Approved	T	T	T	T	F
Condition 2	OTP (One Time Password) Matched	T	T	F	F	X
Condition 3	Sufficient Money in the Account	T	F	T	F	X
Action 1	Transfer Money	Execute				
Action 2	Show a Message as 'Insufficient Amount'		Execute			
Action 3	Block The Transaction Incase of Suspicious Transaction			Execute	Execute	X

DECISION TABLE TESTING – EXAMPLE 4.3

A supermarket has a loyalty scheme that is offered to all customers. Loyalty cardholders enjoy the benefits of either additional discounts on all purchases (rule 3) or the acquisition of loyalty points (rule 4), which can be converted into vouchers for the supermarket or to equivalent points in schemes run by partners. Customers without a loyalty card receive an additional discount only if they spend more than £100 on any one visit to the store (rule 2), otherwise only the special offers offered to all customers apply (rule 1).

	Rule 1	Rule 2	Rule 3	Rule 4
Conditions:				
Customer without loyalty card	T	T	F	F
Customer with loyalty card	F	F	T	T
Extra discount selected	–	–	T	F
Spend > £100	F	T	–	–
Actions:				
No discount	T	F	F	F
Extra discount	F	T	T	F
Loyalty points	F	F	F	T

Exercise 4.4

A mutual insurance company has decided to float its shares on the stock exchange and is offering its members rewards for their past custom at the time of flotation. Anyone with a current policy will benefit provided it is a 'with-profits' policy and they have held it since 2001. Those who meet these criteria can opt for either a cash payment or an allocation of shares in the new company; those who have held a qualifying policy for less than the required time will be eligible for a cash payment but not for shares. Here is a decision table reflecting those rules.

Exercise 4.4-- Solution

	Rule 1	Rule 2	Rule 3	Rule 4
Conditions:				
Current policy holder	Y	Y	Y	N
Policy holder since 2001	N	Y	N	–
'With-profits' policy	Y	Y	N	–
Actions:				
Eligible for cash payment	Y	Y	N	N
Eligible for share allocations	N	Y	N	N

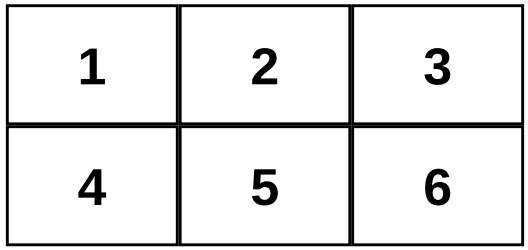
Example

- ***Credit card worked example***
- Let's look at another example. If you are a new customer opening a credit card account, you will get a 15% discount on all your purchases today. If you are an existing customer and you hold a loyalty card, you get a 10% discount. If you have a coupon, you can get 20% off today (but it can't be used with the 'new customer' discount). Discount amounts are added, if applicable.

Conditions	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**	**Rule 5**	**Rule 6**	**Rule 7**	**Rule 8**
New customer (15%)	T	T	T	T	F	F	F	F
Loyalty card (10%)	T	T	F	F	T	T	F	F
Coupon (20%)	T	F	T	F	T	F	T	F
Actions								
Discount (%)	X	X	20	15	30	10	20	0

State Transition Testing

Not covered in this course. Info available as supplement



Dynamic Testing Techniques

Contents

White Box test design and measurement techniques

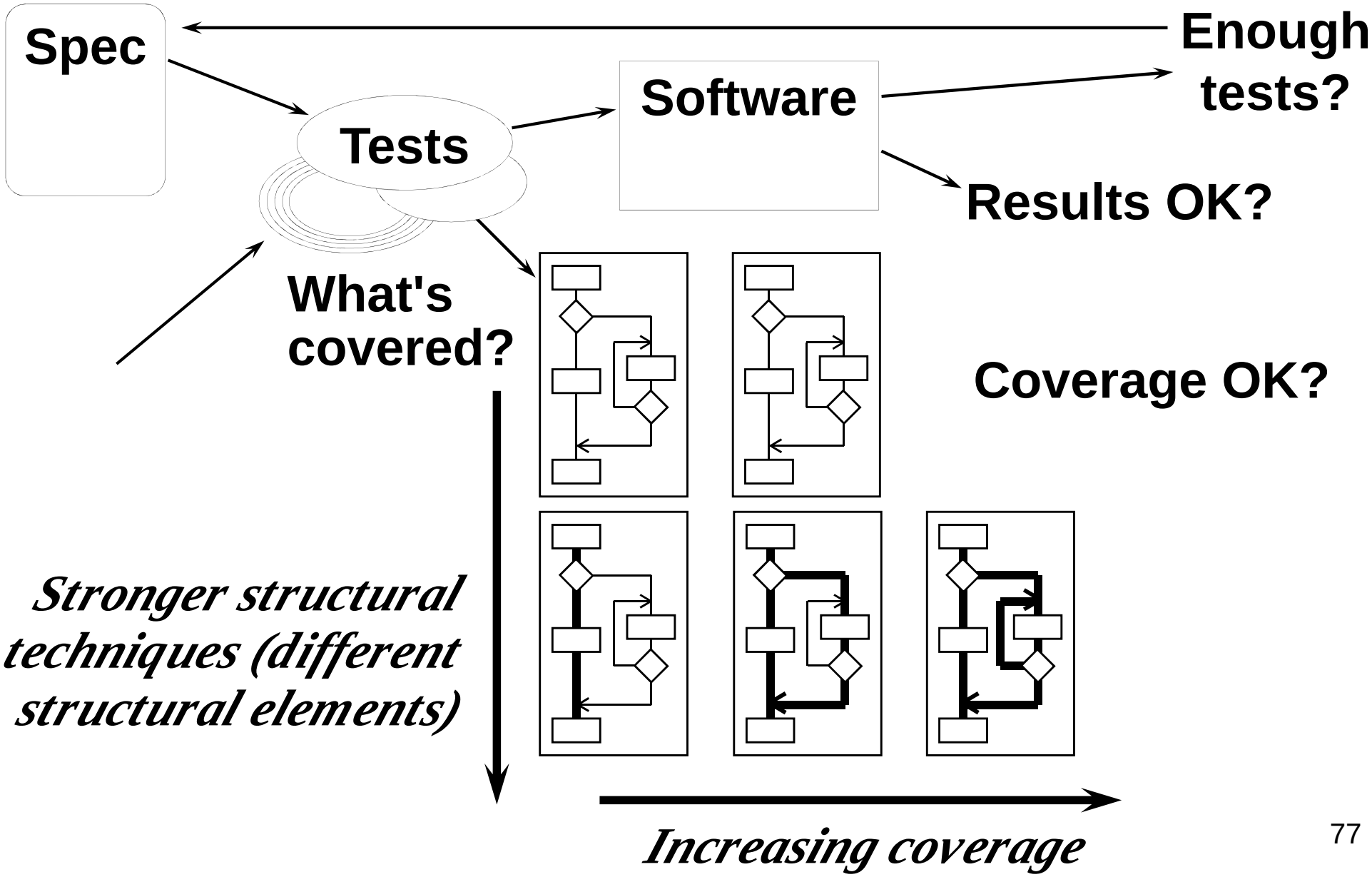
- **Techniques defined in BS 7925-2**
  - Statement testing ✓
  - Branch / Decision testing ✓
  - Data flow testing ✓
  - Branch condition testing ✓
  - Branch condition combination testing ✓
  - Modified condition decision testing ✓
  - LCSAJ testing ✓
- **Also defines how to specify other techniques**

Also a measurement technique?

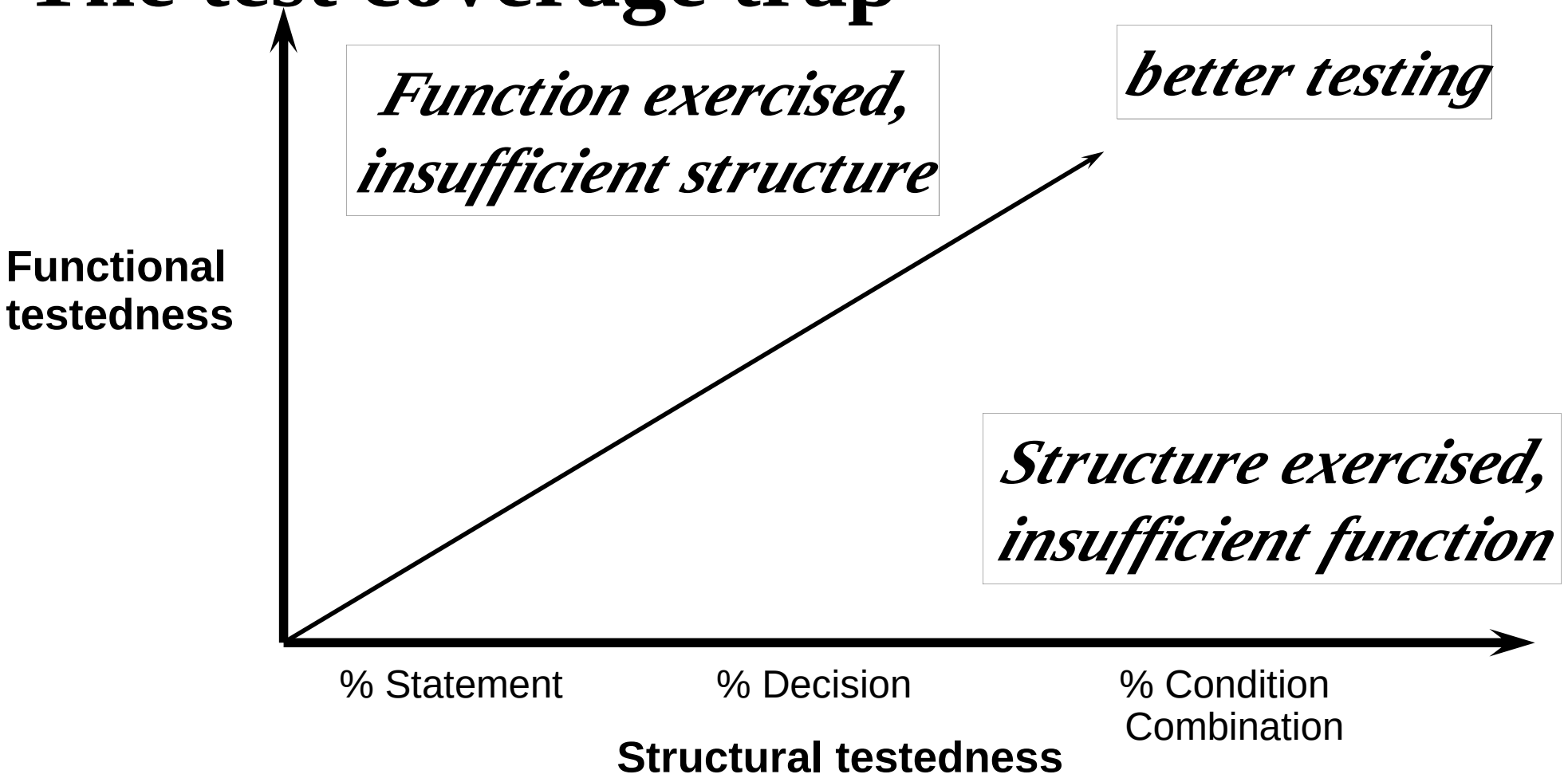
✓ **= Yes**

✗ **= No**

Using structural coverage



The test coverage trap



100% coverage does not mean 100% tested!

Coverage is not Thoroughness

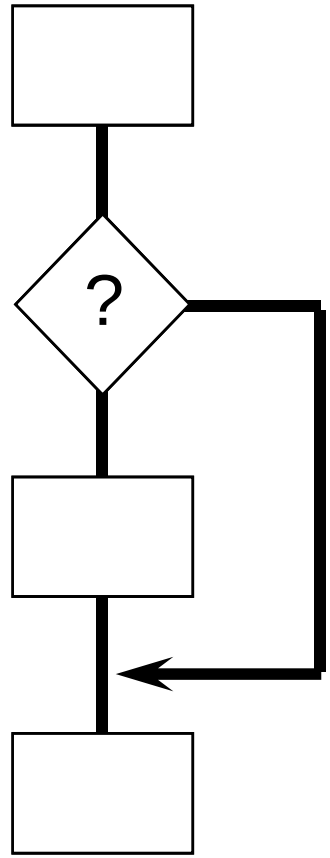
Statement coverage

Statement coverage is normally measured by a software tool.

- **percentage of executable statements exercised by a test suite**

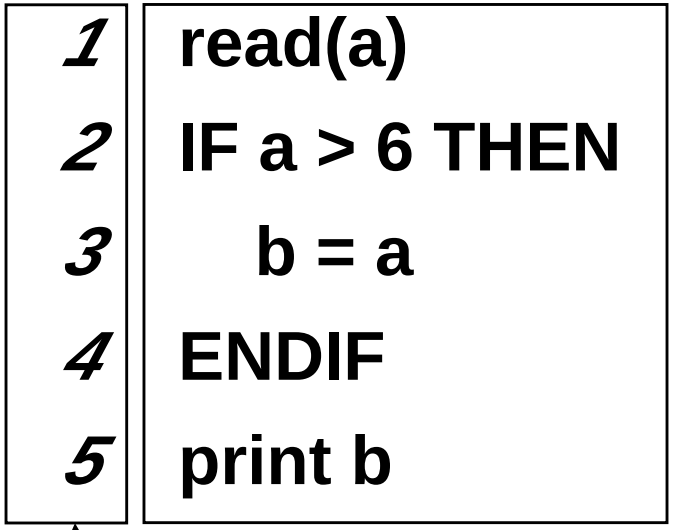
$$= \frac{\text{number of statements exercised}}{\text{total number of statements}}$$

- **example:**
  - program has 100 statements
  - tests exercise 87 statements
  - statement coverage = 87%



Typical ad hoc testing achieves 60 - 75%

Example of statement coverage



Test case	Input	Expected output
1	7	7

Statement numbers

As all 5 statements are ‘covered’ by this test case, we have achieved 100% statement coverage

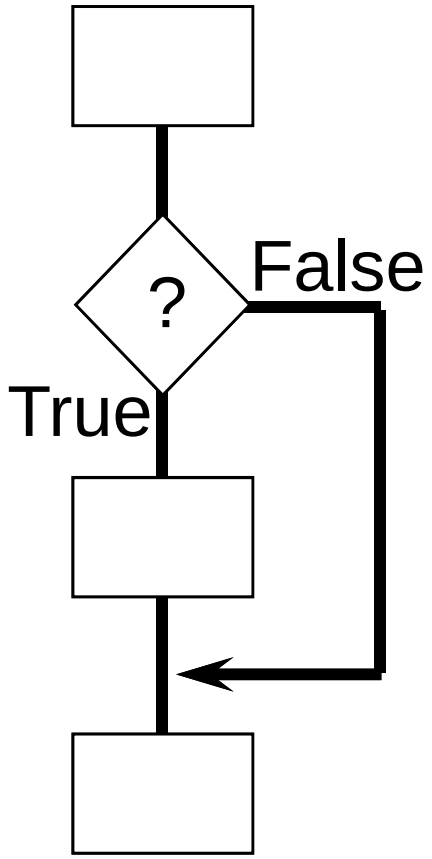
Decision coverage (Branch coverage)

Decision coverage is normally measured by a software tool.

- **percentage of decision outcomes exercised by a test suite**

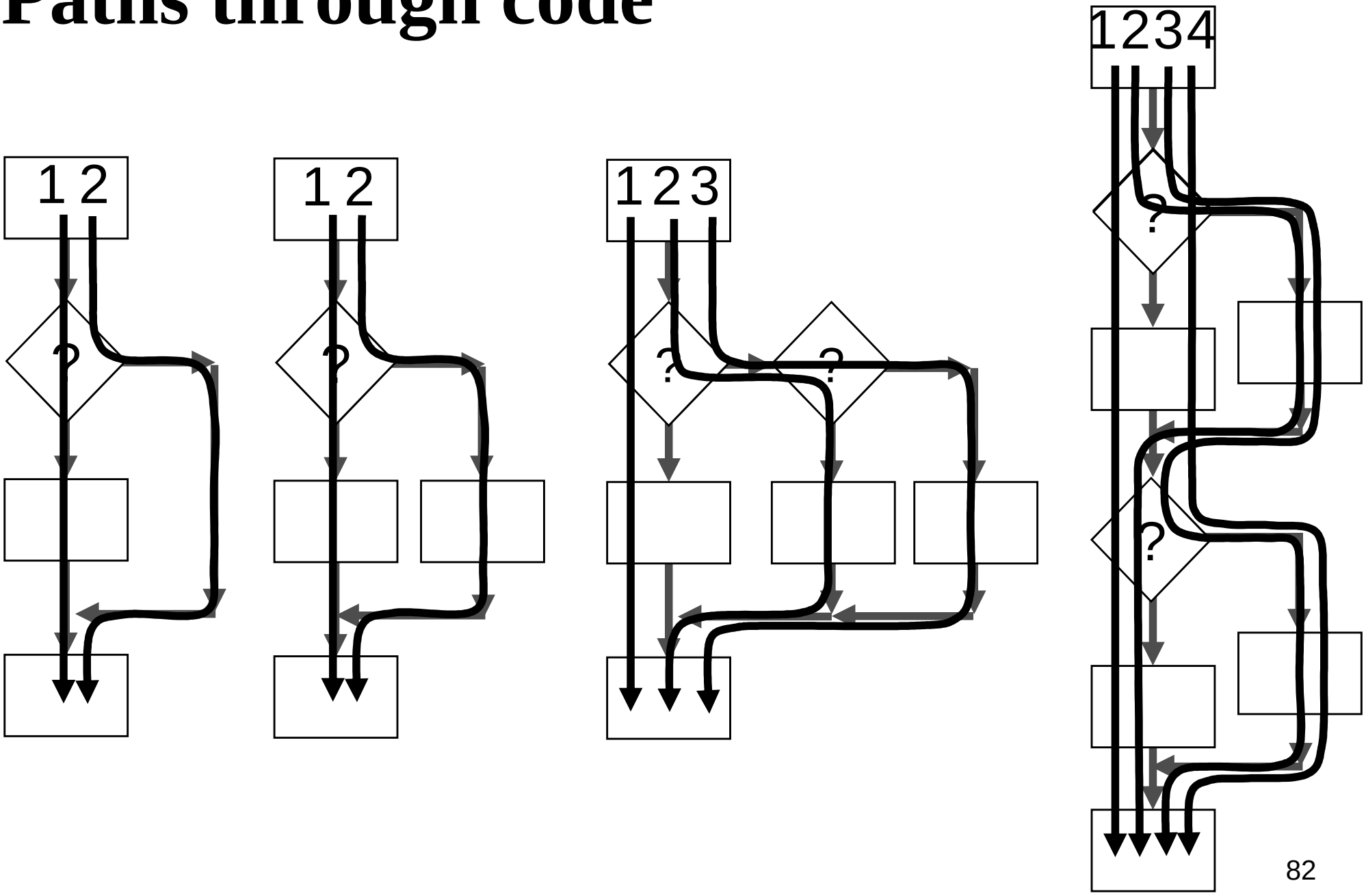
$$= \frac{\text{number of decisions outcomes exercised}}{\text{total number of decision outcomes}}$$

- **example:**
  - program has 120 decision outcomes
  - tests exercise 60 decision outcomes
  - decision coverage = 50%

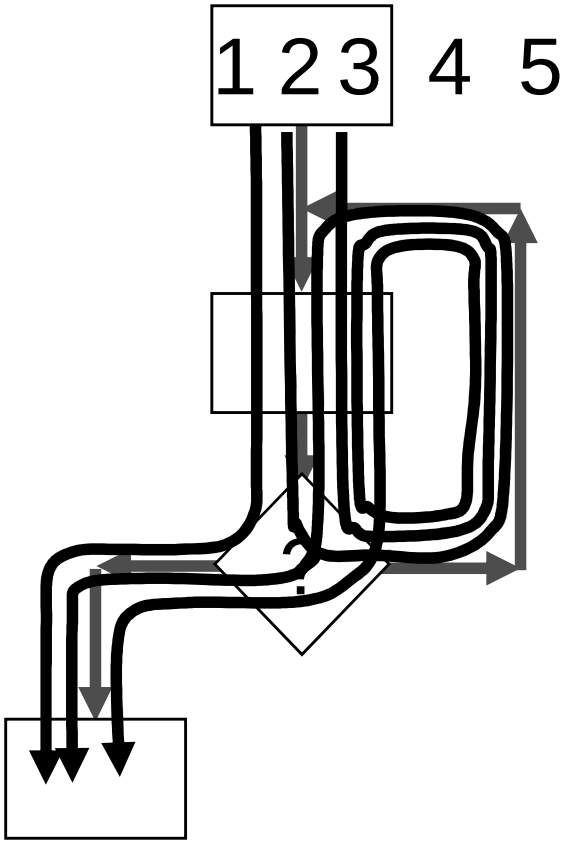


Typical ad hoc testing achieves 40 - 60%

Paths through code



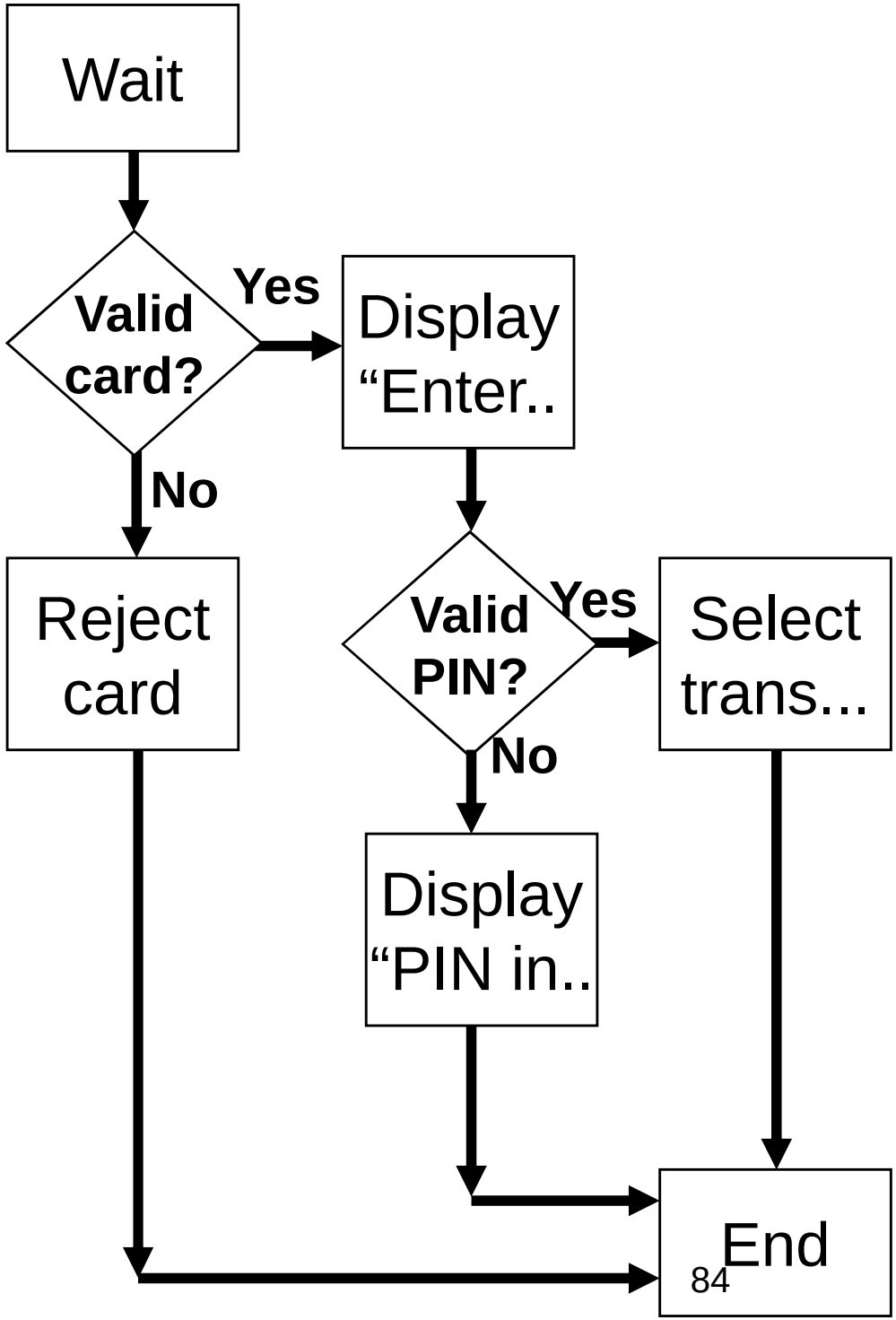
Paths through code with loops



for as many times as it is possible to go round the loop (this can be unlimited, i.e. infinite)

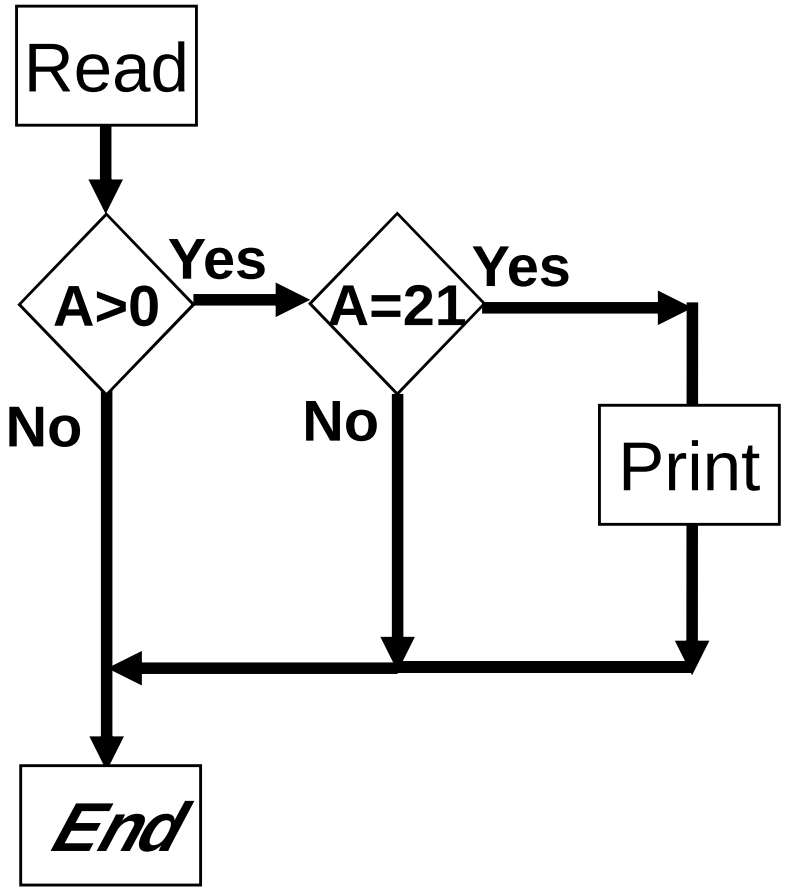
Example 1

Wait for card to be inserted
IF card is a valid card THEN
display “Enter PIN number”
IF PIN is valid THEN
select transaction
ELSE (otherwise)
display “PIN invalid”
ELSE (otherwise)
reject card
End



Example 2

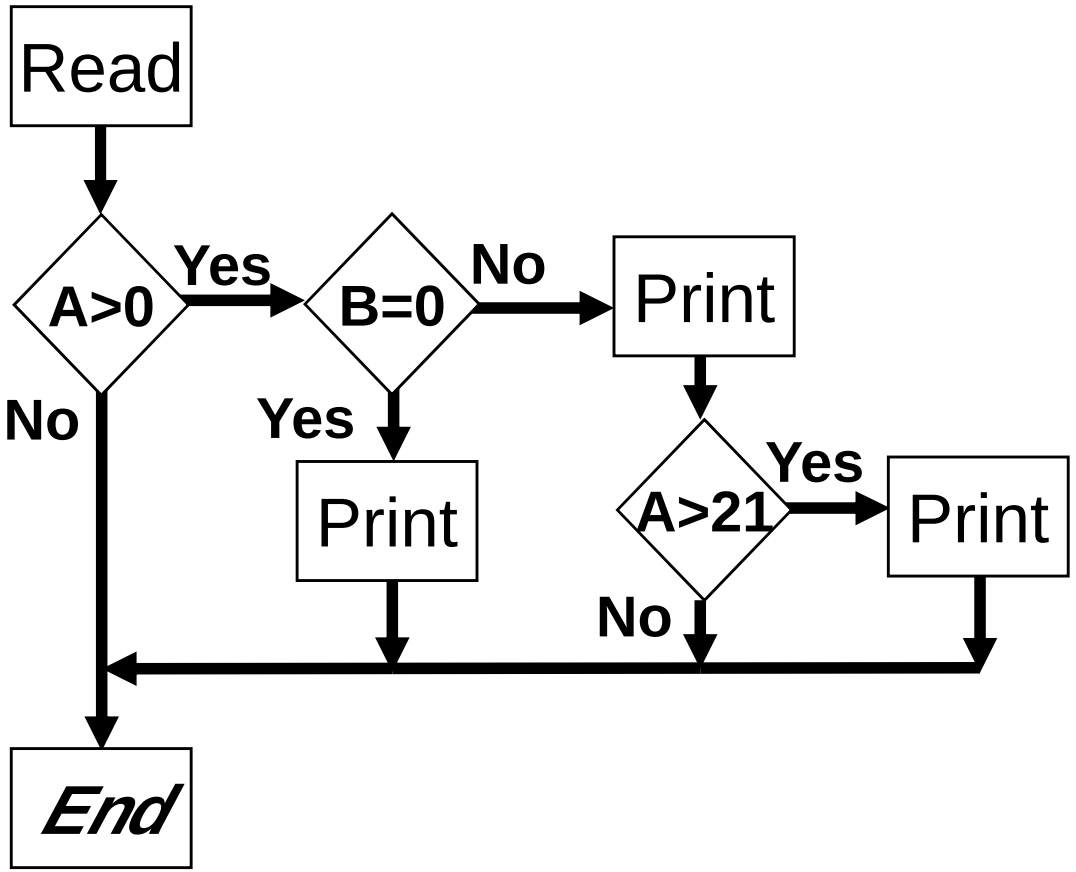
```
Read A
IF A > 0 THEN
    IF A  = 21 THEN
        Print “Key”
    ENDIF
ENDIF
```



- Cyclomatic complexity: _____
- Minimum tests to achieve:
  - Statement coverage: _____
  - Branch coverage: _____

Example 3

1. Read A
2. Read B
3. IF A > 0 THEN
4. IF B = 0 THEN
5. Print “No values”
6. ELSE
7. Print B
8. IF A > 21 THEN
9. Print A
10. ENDIF
11. ENDIF
12. ENDIF



- Cyclomatic complexity: _____
- Minimum tests to achieve:
  - Statement coverage: _____
  - Branch coverage: _____

Read A= 22, B =0→ 1,2,3,4,5,,12
Read A = 22, B =1→ 1,2,3,4,6,7, 8, 9, 10, 11, 12

Example 4

1. Read A
2. Read B
3. IF A < 0 THEN
4. Print “A negative”
5. ELSE
6. Print “A positive”
7. ENDIF
8. IF B < 0 THEN
9. Print “B negative”
10. ELSE
11. Print “B positive”
12. ENDIF

A= -1, B =1→1,2,3,4, 7,8, 10, 11, 12
A = 1, B = -1→ 1,2,3, 5, 6,7, 8, 9, 12

- Cyclomatic complexity: _____
- Minimum tests to achieve:
  - Statement coverage: _____
  - Branch coverage: _____

Example 5

```
Read A
Read B
IF A < 0 THEN
    Print  “A negative”
ENDIF
IF B < 0 THEN
    Print  “B negative”
ENDIF
```

- Cyclomatic complexity: _____
- Minimum tests to achieve:
  - Statement coverage: _____
  - Branch coverage: _____

Example 6

```
Read A
IF A < 0 THEN
    Print  “A negative”
ENDIF
IF A > 0 THEN
    Print  “A positive”
ENDIF
```

- Cyclomatic complexity: _____
- Minimum tests to achieve:
  - Statement coverage: _____
  - Branch coverage: _____

STATEMENT TESTING – EXAMPLE 4.6

Here is an example of the kind you might see in an exam. Try to answer the question, but if you get stuck the answer follows immediately in the text.

Here is a program. How many test cases will you need to achieve 100 per cent statement coverage and what will the test cases be?

```
1   Program BestInterest
2   Interest, Base Rate, Balance: Real
3
4   Begin
5   Base Rate = 0.035
6   Interest = Base Rate
7
8   Read (Balance)
9   If Balance > 1000
10  Then
11      Interest = Interest + 0.005
12      If Balance < 10000
13      Then
14          Interest = Interest + 0.005
15      Else
16          Interest = Interest + 0.010
17      Endif
18  Endif
19
20  Balance = Balance × (1 + Interest)
21
22  End
```

Exercise 4.7
For the following program:

```
1   Program Grading
2
3   StudentScore: Integer
4   Result: String
5
6   Begin
7
8   Read StudentScore
9
10  If StudentScore > 79
11  Then Result = "Distinction"
12  Else
13      If StudentScore > 59
14      Then Result = "Merit"
15      Else
16          If StudentScore > 39
17          Then Result = "Pass"
18          Else Result = "Fail"
19          Endif
20      Endif
21  Endif
22  Print ("Your result is", Result)
23  End
```

- Suppose we ran two test cases, as follows:
- Test Case 1 StudentScore = 50
- Test Case 2 StudentScore = 30

- (1) Would 100% **statement coverage** be achieved?

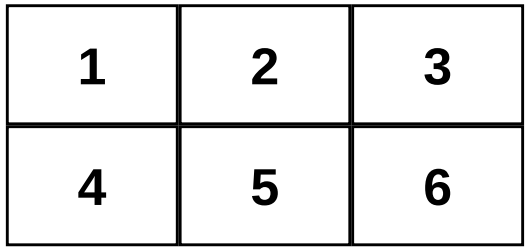
- (2) If not, which lines of pseudo code will not be exercised?

DECISION TESTING – EXAMPLE 4.7

Let us try an example without a loop now.

```
1 Program Age Check
2
3 CandidateAge: Integer;
4
5 Begin
6
7 Read(CandidateAge)
8
9 If CandidateAge < 18
10 Then
11          Print (“Candidate is too young”)
12 Else
13          If CandidateAge > 30
14          Then
15                    Print (“Candidate is too old”)
16          Else
17                    Print(“Candidate may join Club 18–30”)
18          Endif
19 Endif
20
21 End
```

how many test cases are needed for 100 per cent decision coverage and see if you can identify suitable test cases.



Dynamic Testing Techniques

Contents

Non-systematic test techniques

- **Trial and error / Ad hoc**
- **Error guessing / Experience-driven**
- **User Testing**
- **Unscripted Testing**

A testing approach that is only rigorous, thorough and systematic is incomplete

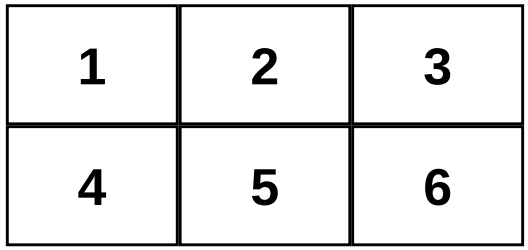
Error-Guessing

- **always worth including**
- **after systematic techniques have been used**
- **can find some faults that systematic techniques can miss**

Not a good approach to start testing with

Error Guessing: deriving test cases

- **Consider:**
  - past failures
  - intuition
  - experience
  - brain storming
  - “What is the craziest thing we can do?”



Dynamic Testing Techniques

Summary: Key Points

Test techniques are ‘best practice’: help to find faults

Black Box techniques are based on behaviour

White Box techniques are based on structure

Error Guessing supplements systematic techniques