Test Answers

Paper – 2017

2017 - 1 (i) Branch coverage subsumes statement coverage because if all the conditional statements are evaluated to true then every statement is executed. If every branch is covered then every statement must also be covered but not vice versa.

In the given example,

While performing branch coverage if we consider x = 0 and y = 0 and z = 0 and x = 0 and y = 0 and z = 1 every statement of the code snippet is executed thus ensuring 100% statement coverage even though statement coverage is not explicitly carried out

2017 - 1. (ii) The fault revealing effectiveness of C2 is much better than C1 because C2 only requires half the test cases to test get full coverage.

2017 - 1. (iii)(a)

x : {0, 1}
y : {-1, 0}

	(x=0, y=-1)	(0,0)	(1,-1)	(1,0)
If-0	F	F	T	T
If-1	F	F	F	T
If-2	F	T	T	F
returns	2	3	3	1

2017 - 1. (iii)(b)

There are a minimum of 3 test cases that will cover all 3 feasible branches testing for both True and False conditions of all 3 branches:

1. For x = 1 and y = 0
Branch: if-0 evaluates to true
Branch: if-1 evaluates to true
Branch: if-2 isn't executed

2. For x = 1 and y = -1
Branch: if-0 evaluates to true
Branch: if-1 evaluates to false

Branch: if-2 evaluates to true

3. For x = 0 and y = -1
Branch: if-0 evaluates to False
Branch: if-1 isn't executed
Branch: if-2 evaluates to False

2017 - 1. (iii)(c)
For output : 1
x = 1, y=0
Therefore partition is x=[1] and y=[0]

For output : 2
x = 0 and y = -1
Therefore partition is x=[0] and y=[-1]

For output : 3
x = 0 and y = 0
and
x = 1 and y = -1
Therefore partition is x=[0,1] and y=[0,-1]

No other output is possible

2017 - 1. (iii)(d)
When x = 1 and y = -1, the original unchanged program will return 3
When x = 1 and y = -1, the mutant program test will return 2

So the answer is x = 1 and y = -1

2017 - 2. (i) The purpose of the formal specification of the program is to clearly specify the domain of the inputs, the expected outputs and any exceptions that should be thrown. It should be very concise. If the formal specification is longer than the program itself then, then either the program is very trivial or the specification is very inefficiently written. The specification should give the user an idea of what the program is about and not delve into unnecessary details.

2017 - 2. (ii) (a) Input variable : k

2017 - 2. (ii) (b) Output variable : x

2017 - 2. (ii) (c) Pre-condition : k = 8 or even number greater than 8

2017 - 2. (ii) (d) Precondition: k = 8 or 10 or 12 or 14 or 16 or 18 or 20 or 22 or 24 or 26 or 28

Paper – 2018

2018 - 1 (i) Branch coverage subsumes statement coverage because if all the conditional statements are evaluated to true then every statement is executed. If every branch is covered then every statement must also be covered but not vice versa.

In the given example,

While performing branch coverage if we consider x = 1 and y = -1 and x = 1 and y = 0 every statement of the code snippet is executed thus ensuring 100% statement coverage even though statement coverage is not explicitly carried out

2018 - 1. (ii) The fault revealing effectiveness of C2 is much better than C1 because C2 only requires half the test cases to test get full coverage.

2018 - 1. (iii)(a)

x : {0, 2}
y : {0, 1, 2}

	(x=0, y=0)	(2,0)	(2,1)	(2,2)
If-0	F	T	T	T
If-1	F	F	T	F
If-2	T	F	F	T
returns	3	2	2	3

2018 - 1. (iii)(b)

There are a minimum of 3 test cases that will cover all 3 feasible branches testing for both True and False conditions of all 3 branches:

1. For x = 2 and y = 1
Branch: if-0 evaluates to true
Branch: if-1 evaluates to true
Branch: if-2 isn't executed

2. For x = 2 and y = 2
Branch: if-0 evaluates to true
Branch: if-1 evaluates to false
Branch: if-2 evaluates to true

3. For x = 0 and y = 1
Branch: if-0 evaluates to False

Branch: if-1 isn't executed
Branch: if-3 evaluates to False

2018 - 1. (iii) (c)

For output : 2
x = 2, y=1
and
x = 0, y =[1,2]
Therefore partition is x=[0,2] and y=[1,2]

For output : 3
x = 0 and y = 0
and
x = 2 and y = [1,2]
Therefore partition is x=[0,2] and y=[0,1,2]

No other output is possible

2018 - 1. (iii) (d)
When x = 1 and y = 0, the original unchanged program will return 2
When x = 1 and y = 0, the mutant program test will return 3

So answer is x = 1 and y = 0

2018 - 2. (i) The purpose of the formal specification of the program is to clearly specify the domain of the inputs, the expected outputs and any exceptions that should be thrown. It should be very concise. If the formal specification is longer than the program itself then, then either the program is very trivial or the specification is very inefficiently written. The specification should give the user an idea of what the program is about and not delve into unnecessary details.

2018 - 2. (ii) (a) Input variable : m

2018 - 2. (ii) (b) Output variable : x

2018 - 2. (ii) (c) Pre-condition : m = 9 or any number greater than 9

2018 - 2. (ii) (d) Precondition: m = 9 or 10 or 11 or 12 or 13 or 14 or 15 or 16 or 17 or 18 or 19

Paper – 2016

2016 1. (i)
Reliability and Safety are different.
Reliability implies given the same input to get the same output every time. A reliable system is very predictable. That does not mean it is necessary safe.
Safety takes into account things like risk and the adverse effects of a risk condition not being met.
For e.g., in programming, if a user attempts to divide a number by 0, he/she will get a dive by 0 exception every single time. This means this behavior is reliable. But it most certainly isn't safe. A divide by 0 instruction is bound to trigger an exception which will cause termination of the program/ software being run which may lead to incomplete results which may be unsafe.

2016 1. (ii)
2 reasons:
1. Privacy seems to be less important with time. Technologies seem to operate on more private information as compared to the past. Things like telephone numbers, email ids and pictures are way more accessible now than before
2. Ease of access to information also increases the risk to hacking. Wesites like social media websites contain huge amounts of personal data of millions of people. This ease of access to important data is a major incentive to potential hackers to try and compromise a system and gain unethical access to private data.

2016 1. (iii) (a) No C.F is not a minimal cut set. Only C is a minimal cut set.

If F operates fine and C fails -> A fails
If F fails and C fails -> A fails

So it is clear that C determines A failing or not.

2016 1. (iii) (b) Minimal cut sets:

(C) is the only minimal cut set

2016 1. (iii) (c) C is more critical than any other component. From the fault tree it is clear that if B, D, E, F, G all work fine but C fails, then C is also guaranteed to fail.

If every component apart from F work fine but F fails A still works fine.
If every component apart from G work fine but G fails A still works fine.

C is the most critical.

2016 2. (i) (a)
Refer diagram sent in chat

2016 2. (ii) (a)
Input variable : j

2016 2. (ii) (b)
Pre-condition : j should be 6 or any number greater than 6

2016 2. (ii) (c)
Pre-condition : j can be 6 or 7 or 8 or 9 or 10 or 11 or 12 or 13 or 14 or 15 or 16

6 >=j <= 16

2016 3. (i) (a) Software testing is undecidable because it is impossible to correctly predict every single fault that may occur in a system. Many faults, especially hardware faults are impossible to prepare test cases for.

2016 3. (i) (b) Practically this means that building software testing tools is very hard especially when it comes to building tools that simulate or replicate hardware failures which may cause things like bit flips, overheating issues in a CPU, etc.

2016 3. (ii) (a)

Input domain for x = 0 and 1
Input domain for y = 0 and -1

	(x=0, y=-1)	(0,0)	(1,-1)	(1,0)
If-0	T	T	F	F
If-1	F	T	F	F
If-2	F	F	T	F
returns	2	0	3	2

2016 3. (ii) (b)

There are a minimum of 3 test cases that will cover all 3 feasible branches testing for both True and False conditions of all 3 branches:

1. For x = 0 and y = 0
Branch: if-0 evaluates to true
Branch: if-1 evaluates to true
Branch: if-2 isn't executed

2. For x = 0 and y = -1

Branch: if-0 evaluates to true
Branch: if-1 evaluates to false
Branch: if-2 evaluates to false

3. For x = 1 and y = -1
Branch: if-0 evaluates to False
Branch: if-1 isn't executed
Branch: if-3 evaluates to True

2016 3. (ii) (c)
For output : 0
x = 0 and y = 0
Therefore partition is x=[0] and y=[0]

For output : 2
x = 0 and y =-1
or x = 1 and y = 0
Therefore partition is x=[0, 1] and y=[-1, 0]

For output : 3
x = 1 and y = -1
Therefore partition is x=[1] and y=[-1]

2016 3. (ii) (d)
When x = 1 and y = -1, the original unchanged program will return 3
When x = 1 and y = -1, the mutant program test will return 2

So the answer is x = 1 and y = -1