浙江大学 2017－2018 学年 winter (2) 学期

《Software Testing and Quality Assurance》

课程期末考试试卷

课程号：21120550，开课学院：_Computer Science

考试试卷：A 卷、B 卷（请在选定项上打√）

考试形式：闭、开卷√（请在选定项上打√），

允许带_Textbook or Printed slides of lecture notes only, _入场

考试日期：2018 年 1 月 10 日,考试时间：90 分钟

诚信考试，沉着应考，杜绝违纪。

考生姓名：_____学号：_____所属院系：_____

题序	一	二	三	四	五	六	七	八	总 分
得分									
评卷人									

Answer three questions only, Question one is compulsory so that all students must attempt it. Answer the remaining two questions from any combination from Questions 2-4.

Question 1 [50 marks in total]

(a) The specification for a software program that computes the ticket cost of a concert at a particular venue is given below. The fee is based on the age of the attendee and their membership status. People outside the age range of 18 to 50 are not allowed to buy a ticket.

If the age of the member is from 18 to 35 (inclusive) the ticket cost is €40 if they are not a member and it is €35 if they are a member

If the age of the member is from 36 to 50 (inclusive) the ticket cost is €65 if they are not a member and it is €55 if they are a member.

The program input consists of 2 parameters: int age and boolean isMember. The program output is the int ticketCost value in Euros. Any invalid input returns a value of -€1.

Considering the Black-Box Software testing methods of Equivalence Partitions for (等价类划分)this program fill in table with the input and output partitions. Then write out suitable tests ensuring to outline the partitions covered with each set of test data and highlight the error cases with an asterisk. [23 marks]

[11 marks for filing out the partitions table correctly (1 marks for each entry) and then 12 marks (2 marks each) for filling out each row of the test data table correctly. Note: Input test data numerical values are arbitrary and are picked so that specific text cases are executed]

Test Cases	**Input Partitions**	**Range**
	Parameter - age	
EP1*		Integer.MIN_VALUE…17
EP2		18…35
EP3		36…50
EP4*		51…Integer. MAX_VALUE
	Parameter – isMember	
EP5		True
EP6		False
	Output Partitions	
	Parameter - ticketCost	
EP7*		-1
EP8		35
EP9		40
EP10		55
EP11		65

Test Data

Test No.	Test Cases Covered	Inputs		Expected Outputs
		age	*isMember*	*ticketCost*
1	EP2, 5, 8	25(18~35 之间任意值)	True	35
2	EP2, 6, 9	25(18~35 之间任意值)	False	40
3	EP3, 5, 10	43(36~50 之间任意值)	True	55
4	EP3, 6, 11	43(36~50 之间任意值)	False	65
5	EP1*, 7*	10(<18 任意值)	False	-1
6	EP4*, 7*	60(>50 任意值)	False	-1

(b) Using the Causes and Effects as given below, complete the Truth Table underneath for the black box technique of Combinational Testing. Then redraw the truth table to eliminate 删除 any impossible test cases. Following this, create suitable Test Data.

[15 marks] ~~[1 mark each for getting each column correct in the final truth table, 3 marks for identifying and removing the two impossible test cases~~ **[1 mark for each correct Rule/TestCase to give a total of 8 marks; one mark for the identification/removal of the impossible cases; 1 mark each for each row of test data. Note: Input Test data numerical values are arbitrary and are picked so that specific text cases are executed]**

Causes	**Effects**
18<=age<=35	ticketCost=35
36<=age<=50	ticketCost=40
isMember	ticketCost=55
	ticketCost=65
	ticketCost=-1

First truth table

	Rules							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Causes								
18<=age<=35	T	T	T	T	F	F	F	F
36<=age<=50	T	T	F	F	T	T	F	F
isMember	T	F	T	F	T	F	T	F
Effects			F	F	F	F	F	F
ticketCost=35			T	F	F	F	F	F
ticketCost=40			F	T	F	F	F	F
ticketCost=55			F	F	T	F	F	F
ticketCost=65			F	F	F	T	F	F
ticketCost=-1			F	F	F	F	T	T

Rules 1 and 2 are impossible test cases and are highlighted by the grey shading. They can therefore be removed.

Final truth table

	Rules					
	1	**2**	**3**	**4**	**5**	**6**
Causes						
18<=age<=35	**T**	**T**	**F**	**F**	**F**	**F**
36<=age<=50	**F**	**F**	**T**	**T**	**F**	**F**
isMember	**T**	**F**	**T**	**F**	**T**	**F**
Effects	**F**	**F**	**F**	**F**	**F**	**F**
ticketCost=35	**T**	**F**	**F**	**F**	**F**	**F**
ticketCost=40	**F**	**T**	**F**	**F**	**F**	**F**
ticketCost=55	**F**	**F**	**T**	**F**	**F**	**F**
ticketCost=65	**F**	**F**	**F**	**T**	**F**	**F**
ticketCost=-1	**F**	**F**	**F**	**F**	**T**	**T**

Test Data

Test No.	**Test Cases Covered**	**Inputs**		**Expected Outputs**
		age	*isMember*	*ticketCost*
1	TT 1	**25(18~35 之间任意值)**	**True**	**35**
2	TT 2	**25(18~35 之间任意值)**	**False**	**40**
3	TT 3	**43(36~50 之间任意值)**	**True**	**55**
4	TT 4	**43(36~50 之间任意值)**	**False**	**65**
5	TT 5	**5(<18 任意值)**	**True**	**-1**
6	TT 6	**55(>50 任意值)**	**False**	**-1**

(c) This program 'PhoneIns' computes the cost of a smartphone insurance policy and outputs a value for the premium as denoted by *p*. It takes two inputs of *age* and *OS* (Operating System) type.
If the age entered is less than 16 or greater than 99 the program returns a premium of zero, *p*=0. The input for OS takes the form of 'I' for IoS, 'A' for Android, and 'W' for Windows. If an incorrect value for the OS is entered, the program returns *p*=0.

In general the insurance premium is €50, *p*=50.

However, if a person has an iPhone and is under 25 then an extra €25 is added to the premium, *p*=75.

If the person is aged between 40 and 60 (inclusive) and they have an Android phone the premium falls by €10, *p*=40.

If the person is aged between 61 and 65 inclusive the premium falls by €5, *p*=45.

Line No.	Code
1	`public int phoneIns (int age, char OS) {`
2	`int p;`
3	`if ((age<16) || (age>99) || (OS!='I' &&OS!='A' &&  OS!='W'))`
4	`p=0;`
5	`else {`
6	`    p=50;`
7	`    if ((age<25) && (OS=='I'))`
8	`    p += 25;`
9	`    else {`
10	`        if ((age>=40) && (age<=60) && OS=='A')`
11	`        p -= 10;`
12	`        else if ((age>=61) && (age<=65))`
13	`        p -= 5;`
14	`    }`
15	`}`
16	`return p;`
17	`}`

Complete the Control Flow Graph (CFG) for the program PhoneIns() using the source code provided. Then, derive test cases and test data for the White Box Software testing method of Statement Testing. Ensure to show the nodes covered by each set of test data [9 marks]

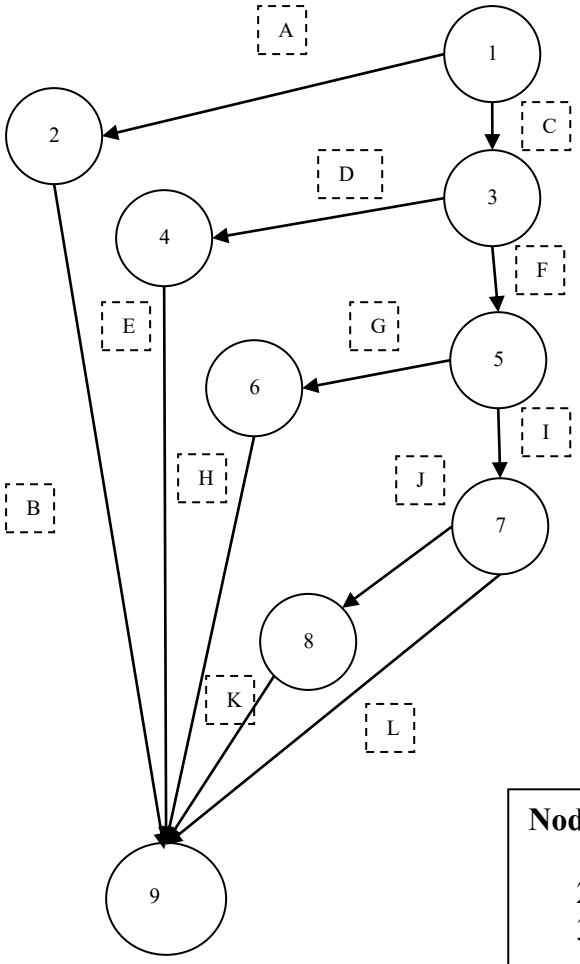
What would you need to add to the test data to satisfy Branch Coverage? Complete the table for Branch Testing, also showing the Branches covered by the test data [3 marks]

[5 marks for the control flow graph and 4 marks for the Statement Coverage test data, 3 marks for correctly filling in the Branch Testing Test Data in the table]

Each node in the CFG is a test case.

[5 marks, For the CFG there is 1 mark for each set of arrows (branches) correctly drawn on the CFG that are correct, that is,
A->B
C->D->E
C->F->G->H
C->F->I->J->K
C->F->I->L
]

The tests cases and test data are **[4 marks/1 mark for each set of cases covered and test data values. Input Test data numerical values are arbitrary and are picked so that specific text cases are executed]**



CFG

PhoneIns

Node	Lines
1	1…3
2	4
3	5, 6, 7
4	8
5	9, 10
6	11
7	12, 13, 14, 15
8	16,…17

Test No.	Test Cases Covered	Inputs		Expected Outputs
		Age	*OS*	*p*
1	SC1, 2, 8	15	I	0
2	SC[1], 3, 4, [8]	20	I	75
3	SC[1], [3], 5, 6, [8]	50	A	40
4	SC[1], [3], [5], 7, [8]	63	W	45

Test No.	Test Cases Covered	Inputs		Expected Outputs
		Age	*OS*	*p*
1	BC A, B	15	I	0
2	BC C, D, E	20	I	75
3	BC[C], F, G, H	50	A	40
4	BC[C], [F], I, J, K	63	W	45
5	BC[C], [F], [I], L	40	W	50

[For the Branch Testing there are 2 marks for Test No. 1-4 as they are the same as for Statement Coverage and 1 mark for Test No. 5]

Only do two Questions from the following [25 marks each]:

Question 2

(a) "The objective of software testing is all about finding software faults". Can you list and explain 3 categories of software faults? [9 marks]

i. Documentation faults
ii. Stress or overload faults
iii. Throughput or performance faults
iv. Recovery faults
v. Syntax Faults

[Only three of the following are required. 3 marks each to give a total of 9 marks]

Software Fault	Description
Syntax Fault	These are due to the nonconformity 不一致 of programming language,
Documentation Faults	Incomplete or incorrect documentation will lead to Documentation faults
Stress of Overload faults 压力过载故障	These happen when data structures are filled past their specific capacity whereas the system characteristics are designed to handle no more than a maximum load planned under the requirements
Throughput or performance faults 吞吐量和性能故障	This is when the developed system does not perform at the speed specified under the stipulated 规定的 requirements

Recovery faults	This happens when the system does not recover to the expected performance even after a fault is detected and corrected 当系统无法恢复到预期的性能时

(b) Concisely 简明 describe three approaches to Debugging software. [3 marks]
[1 mark for each of the following]

Brute Force 蛮力 – hack away at the code from the beginning to the end until the bug is found,
Backtracking 回溯 – start at the point in the program where the problem occurs and then work your way backwards through the logic to determine the cause of the bug
Cause elimination – hypothesize 假设 about what is causing the bug and input test data to check this, using the results to isolate what might be causing the bug

(c) Describe *two* significant differences 显著差异 between Black-box testing and White-box testing. [**8 marks**]

[4 marks for each pair of differences to add to 8 marks]

Three pairs of differences between White Box and Black Box testing are:

Black box	White box
- Test against the specification - Test cases derived from specification - Tests can be reused if the code is updated or additional functionality is added	- Test against the implementation - Test cases derived from the source code - Tests are generally invalidated by any changes to the code

(d) Briefly explain the difference between Unit and System testing. [**5 marks**]

[2.5 marks for each correct answer. Answer for each one need only be two lines long and contain only the essential material of each sample answer here.]

Unit Testing An individual unit of software is tested to make sure it works correctly. This unit may be a single component, or a compound component formed from multiple individual components. Unit testing almost invariably 总是 makes use of the programming interface of the unit.

System Testing The entire software system is tested as a whole to make sure it works correctly. The testing uses the system interface: this may be a Graphical

User Interface for an application (GUI), web interface for a web-server, network interface for a network server, etc.

Question 3

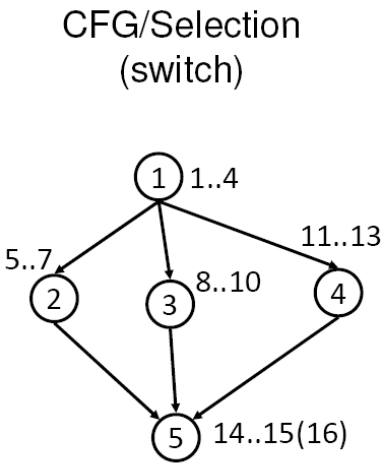
(a) Deciding when the testing of a piece of software should finish can be difficult to judge. Can you suggest two different criteria 标准 or benchmarks 标杆 that could be useful to a Software Testing manager when making that decision? [**5 marks**]

[2.5 marks for each criterion, only two are required]

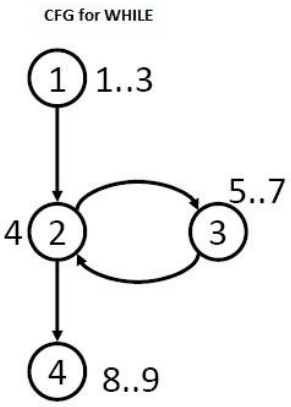
1)A budgetary criteria 预算标准: when the time or budget allocated has expired
2)An activity criteria: when the software has passed all of the planned tests
当软件已经通过了所有的计划测试。
3)A risk management criteria 风险管理标准: when the predicted failure rate meets some quality criteria 当预测的失败率达到一定的质量标准。

(b) What is a control flow graph (CFGs)? Draw sample CFGs illustrating their appearance for a Switch statement and a While loop. **[5 marks]**
[1 mark for the definition of a control flow graph and 2 marks each for the CFG for a Switch statement and a While loop respectively]

A Control Flow Graph (CFG) is a directed graph 有向图 made of vertices 顶点 and edges. The vertices are nodes that represent a basic block of one or more statements.The edges are directed and the direction indicates the flow of control, showing the outcomes of a decision. **[1 marks]**



[2 marks]



[2 marks]

(c) A 'data-flow' approach to testing is a way of looking at a program as a flow of data from one statement to another. The motivation 动机 is to find data flow anomalies 异常. Give explanations of three types of anomalies with examples. [**5 marks]**

[2 marks each for Data Flow anomaly Types 1 and 2, and 1 mark for Data Flow anomaly Type 3]

Type 1: A variable is first defined and then defined again
Example: The second definition of x overrides the first.
x = f1(y);
x = f2(z);
– Four interpretations 解释 of Example 1
The first statement is redundant.
The first statement has a fault -- the intended one might be: w =f1(y).
The second statement has a fault – the intended one might be: v =f2(z).
There is a missing statement in between the two: v = f3(x).

Type 2: A variable is undefined but referenced
Example 2: x = x – y – w;
where w has not been defined by the programmer.
– Two interpretations
The programmer made a mistake in using w.
The programmer wants to use the compiler assigned value of w.

Type 3: Defined but not referenced
Example: Consider x = f(x, y). If x is not used subsequently, we have a Type 3 anomaly.

(d) Give one of the strengths 优势 and one of the weaknesses 劣势 of *five* of the following White-box and Black-box software testing methods **[10 marks]**:

i. Statement Testing
ii. Branch Testing
iii. Path Testing
iv. Equivalence Partitioning
v. Boundary Value analysis
vi. Combinational Testing

2 marks each for just one strength and each weakness for each technique to make a total of 10 marks

Test Type	Strength	Weakness
Statement	1)Provides a minimum level of coverage by executing all statements at least once	1)May be hard to test code that can only be executed in dangerous circumstances 危险环境 2)Will not test unreachable code 3)does not provide coverage for 'NULL else' conditions 不提供"空 else"条件的覆盖率 4)Not demanding of compound

		logical conditions 不要求复合逻辑条件
Branch	1)Branch coverage ensures that each "decision" is tested at least once. 2)100% branch coverage guarantees 100% statement coverage – but the test data is harder to generate.	1)Branch coverage still doesn't exercise either all the reasons for taking each branch, or combinations of different branches taken. 仍然没有充分利用每个分支的原因，或者不同分支的组合。
Path Testing	1)It generates test data in the pattern that data is manipulated in the program rather than following abstract branches 2)A strong form of testing	1)Number of test cases can be very large 2)Difficult to apply it to pointer variables 3)Difficult to apply it to arrays
Equivalence Partitioning	1)Provides a good basic level of testing. 2)Well suited to data processing applications where input variables may be easily identified and take on distinct values. 3)Provides a structured means for identifying basic Test Cases.	1)Correct processing at the edges of partitions is not tested. 2)Combinations of inputs are not tested. 3)The technique does not provide an algorithm for finding the partitions or selecting the test data.
Boundary Value Analysis	Gives more guidance in test data creation Test data focused on areas where faults are more likely to be found	Combinations of input partitions have not been tested 输入分区的组合没有测试。
Combinational Testing	Exercises combinations of test data Expected outputs are created as part of the process	The truth tables can sometime be very large. The solution is to identify sub-problems, and develop separate tables for each. Very dependent on the quality of the specification - more detail means more causes and effects, which takes more time to test; less detail means less causes and effects, but less effective testing

Question 4

(a) State the difference between Software verification 验证 and Software validation 确认 in the context of software testing. [2 marks]

[The definition is worth 2 marks]
Verification is confirming that we have built the software correctly while Validation confirms that we have built the correct software.

[VERIFICATION（验证）：The Purpose of Verification is to ensure that selected work products meet their specified requirements.

VALIDATION（确认）：The Purpose of Validation is to demonstrate that a product or product component fulfills its intended use when placed inits environment.]

验证是指已经实现的软件产品是按照它的需求做的，是符合需求说明书的。

确认是指已经实现的软件产品或产品组件在用户环境下，实现了用户的需要，是符合用户需要的。

(b) Can you give one advantage and one disadvantage of using the V-model for software development? [4 marks]

One advantage from these [2 marks]
It is simple and easy to manage due to the rigidity of the model,

It encourages Verification and Validation at all phases:

Each phase has specific deliverables and a review process.

It gives equal weight to testing alongside development rather than treating it as an afterthought 时候进来的东西.

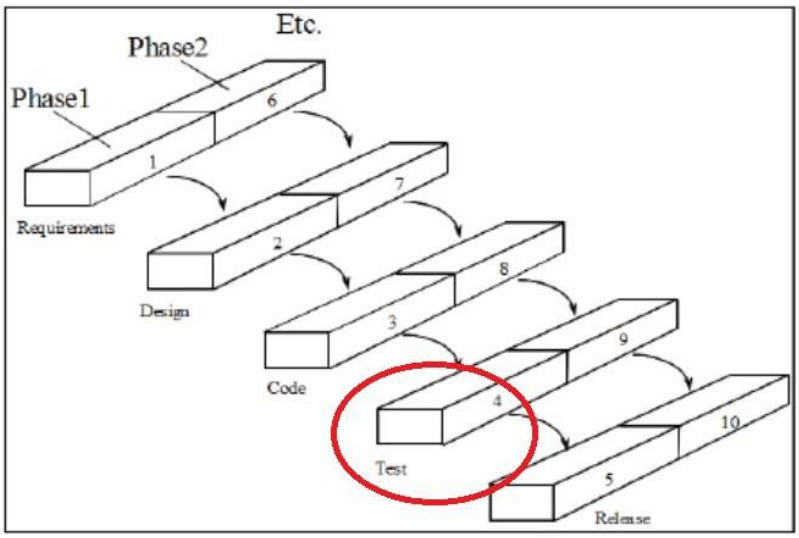
One disadvantage from these [2 marks]
There is no working software produced until late during the life cycle

It is unsuitable where the requirements are at a moderate to high risk of changing.

It is a poor model for long, complex and object-oriented projects

(c) The Incremental development model of Software Development is an 'Agile' method. Draw a diagram that illustrates how the Incremental development model works. Give *one* advantage and *one* disadvantage of it. [7 marks]

The diagram is worth **[3 marks]**



Either of the following is an advantage, only one required [2 marks]
A major advantage of the incremental model is that the product is written and tested in smaller pieces, reducing risk and allowing for change to be included easily

The customer or users is involved from the beginning which means the system is more likely to meet their requirements and they themselves are more committed to the system

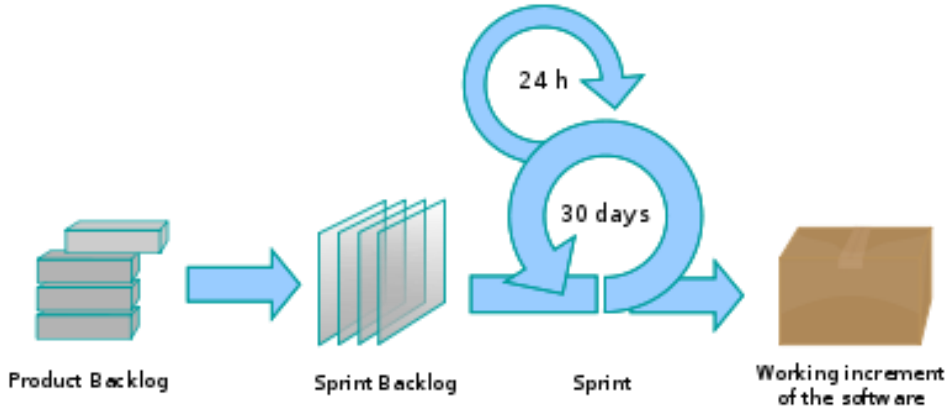
Either of the following is a disadvantage, only one required [2 marks]
It can be difficult to manage because of the lack of documentation in comparison to other models

The continual change to the software can make it difficult to maintain as it grows in size.

(d) For the SCRUM approach to managing software development explain the following two terms. Also, sketch 速画 a diagram to show their relationship to each other. [10 marks] **[4 Marks for the diagram and 3 Marks each for the explanation of the two terms]**

i. Product Backlog
ii. Sprint and Sprint Backlog

The graph should resemble 类似 the graph given below. **[4 marks]**



The answer need contain only two of the points made for each definition below:

Product Backlog [3 marks]
The Product Backlog is a prioritized 优先化 list of all product requirements. The backlog items can come from: Users, customers, sales, marketing, customer service 售后服务, and engineering. [+1]The items most likely to generate value are at the top. These are the most desirable 值得的 features and are whatever needs to be done in order to successfully deliver a working software system. It is divided into proposed releases. [+1]The Product Backlog is never finalized - it emerges 出现 and evolves 发展 with the product [+1]
[product backlog 是一个具有优先级的需求列表， 并对每个需求进行了粗略的估算。在 Scrum 中表示可以预知的所有任务，包括未细化的产品功能要求、Bugs、缺陷、用户提出的改进、具竞争力的功能及技术升级等，按优先级定义出来，这些任务可能不是完整的，甚至可能随时会更改或添加。Prodcut Backlog 永远处于不完整状态，它随着产品及其使用环境的变化而变化，它是动态的，管理层不断对之做出改变，确定产品需求，保证产品适用性、实用性和竞争性。]

Sprint and Sprint Backlog [3 marks]
SCRUM teams take on as much of the product backlog as they think they can turn into an increment of product functionality within a 30-day iteration 迭代. This is called a Sprint. The team maintains a list of tasks to perform during each Sprint that is called the Sprint Backlog.[+1]

(e) Explain one benefit to the DevOps technique. [2 marks]

Any one of these is a benefit [2 marks]
Faster Time to market – reduced cycle times and higher deployment rates

Increased Quality – better availability of the product as it is being created, increased change success rate and fewer failures

Increased organizational effectiveness – more time spent on value adding activities and greater value being delivered to the customer