2022-CS265FZ-January-Solutions

1

▼ **(a) What are the main purposes of software testing ?**

- *Evaluate the quality of a software product whether verify the fulfillment of all specified requirements*
- *Find and prevent defects in the software product*

▼ **(b) A program is designed for an online shop that determines whether a customer should get a price discount on their next purchase, based on their bonus points to date, and whether they are a gold customer. The specification of the program is given below.**

Bonus points accumulate as customers make purchases. The value for bonus points is an integer. Gold customers get a 5% DISCOUNT once they have exceeded 80 bonus points, and a 7% DISCOUNT exceeded 120 bonus points. Other customers only get a 5% DISCOUNT once they have more than 120 bonus points. All customers always have at least one bonus point. The Bonus points parameter is invalid if it is less than 1.

The program returns one of the following:
• FULLPRICE
• 5% DISCOUNT
• 7% DISCOUNT
• ERROR

The inputs are the number of bonusPoints the customer has, and a flag indicating whether the customer is a gold customer or not (true or false, respectively). For the sake of simplicity, we will ignore illegal input values at the moment (where the first parameter is not a number, or the second parameter is not a boolean).

▼ **1) Using the table provide below, develop the test cases and identify the equivalence classes for each input parameter and the expected output.**

Test Case	Parameter	Partition
EC-1	***bonusPoints***	[Integer.Min_value,0]
EC-2		[1,80]
EC-3		[81,120]
EC-4		[121, Integer.Max_value]

EC-5	***goldCustomer***	true
EC-6		false
EC-7	**Return value**	FULLPRICE
EC-8		7% DISCOUNT
EC-9		5% DISCOUNT
EC-10		ERROR

▼ **2) Using the table provide below, identify suitable test data for a test using the Equivalence Class technique.**

Test ID	**Test Cases Covered**	***bonusPoints***	***goldCustomer***	**Return value**
T1.1	EC-1,5,10	0	true	ERROR
T1.2	EC-2,6,7	45	false	FULLPRICE
T1.3	EC-3,[5],8	100	true	7% DISCOUNT
T1.5	EC-4,[5],9	121	true	5% DISCOUNT

2

▼ **(a) What do a Node and an Edge represent in a control flow graph?**

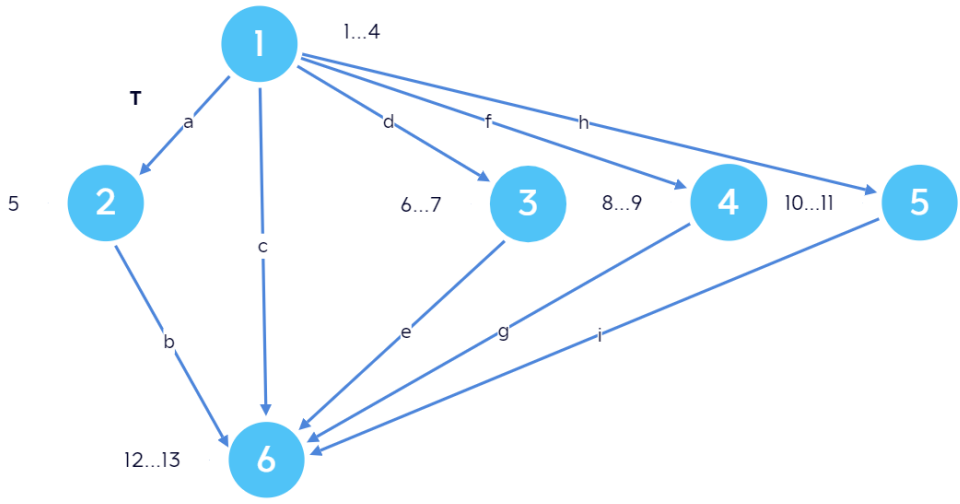
- *A node represents one or more indivisible statements in the source code*
- *Each edge represents a branch*

▼ **(b) Given a program source code as shown below**

```
1   public static boolean checkContainer(containerSize
2   size,int weight,int volume) {
3      boolean ok = true;
4      if ( (weight<0) || (volume<0) )
5         ok = false;
6      else if ( (size==Small) && ((weight>100)||(volume>1)))
7         ok = false;
8      else if (size==Medium)&&((weight>500)||(volume>10)))
9         ok = false;
10     else if ((size==Large)&&((weight>1000)||(volume>20)))
11        ok = false;
12     return ok;
13  }
```

▼ **1) construct a control flow graph for the program.**

`多个 else if 语句，相当于 switch`



▼ **2) identify all the paths for Statement Coverage.**

Path1: 1→6

Path2: 1→2→6

Path3: 1→3→6

Path4: 1→4→6

Path5: 1→5→6

▼ **3) identify all the paths for Branch Coverage.**

Path1: c

Path2: a→b

Path3: d→e

Path4: f→g

Path5: h→i

3

▼ **(a) The goal of Decision Condition Coverage is for every condition and decision to take on the values true and false. Use an example to explain the relationship between a condition and a decision.**

Each condition will be set true or false, thus leading a different decison.

Test Case	Condition/Decision
CDC-1	((passengers > 0) && (passengers <= allSeats))
CDC-2	! ((passengers > 0) && (passengers <= allSeats))
CDC-3	(passengers > 0)
CDC-4	!((passengers > 0)
CDC-5	(passengers <= allSeats)
CDC-6	! (passengers <= allSeats)

▼ (b) In a Path Coverage test, given a control flow graph as shown, use the regular expression method to

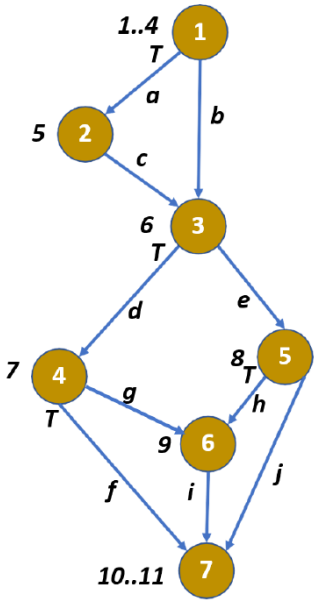


Figure 1

▼ 1) calculate the number of paths (show the step-by-step process).

1.(2+0).(4+5).(6+0).7

1.(1+1).(1+1).(1+1).1=8

▼ 2) identify all the paths (show the step-by-step process

1.2.3.4.6.7

1.2.3.4.7

1.3.4.6.7

1.3.4.7

1.2.3.5.6.7

1.2.3.5.7

1.3.5.6.7

1.3.5.7

4

▼ (a) Early dataflow analyses focused on a set of faults associated with variables, known as Dataflow Anomalies. Give three typical dataflow anomalies.

- *Defined and then defined again*
- *Undefined but referenced*
- *Defined but not referenced*

▼ (b) What are the pros and cons of waiting until all the code has been written and then testing the finished product all at once?

Pros:

- *It may be more efficient to test the entire product at once, rather than testing individual components separately. This can save time and resources that would otherwise be spent on multiple rounds of testing.*

Cons:

- *If there are issues with the code, it may be more difficult to identify the root cause if the entire product is tested at once. This can make it harder to fix the issues and may result in a longer testing process.*

▼ (c) Statement Coverage and Branch Coverage are two basic structural testing techniques. When testing a given program, a test engineer decided to use the Branch Coverage technique to test the program. Would a subsequent test using the Statement Coverage technique add any extra coverage to the overall test result? Give reasons for your answer

Yes. Since the Branch Coverage technique only focuses on branches and not on individual statements, using the Statement Coverage technique would add extra coverage to the overall test result by ensuring that all statements in the program have been executed at least once.