CS265FZ Software Testing
Lab 7 – Path Coverage

There are TWO exercises to be completed.

Two pieces of work need to be submitted:

1. Fill in this lab sheet and submit it to Moodle. You don't need to attach your source code in this form. You need to upload your source code separately.
2. Submit all the required source code to Moodle. Make sure your source code is tested in Eclipse and is executable.

Program 1

Lab7_Program1 is a program for screening orders received from customers based on three values: quantity of the order (***quantity***), credit status of customer (***credit***), and the inventory quantity (***inventory***). The program output is a string. It depends on the values of the three parameters, the output will be: "Accept", "Reject", or "Defer".

The calculation is as follows:

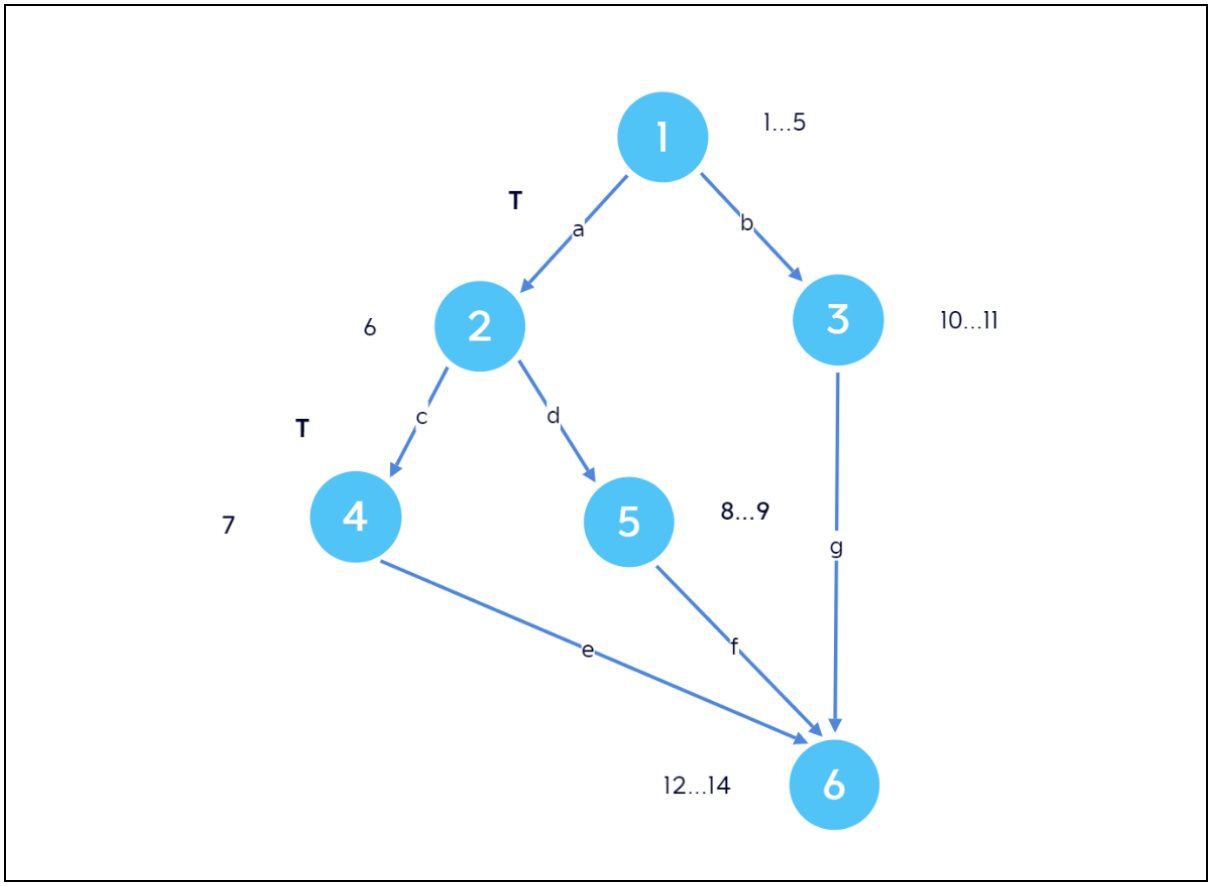
- If the ordered quantity is smaller than or equal to 1000 items (the maximum limit), and the customer is credit-worthy, and the inventory is larger than or equal to the ordered quantity, then the order will be accepted.
- If the ordered quantity is smaller than or equal to 1000 items (the maximum limit), and the customer is credit-worthy, but the inventory is less than the ordered quantity, then the order will be deferred.
- All the other cases, the orders will be rejected.

```
1  package edu.st.lab7;
2  public class Lab7_Program1 {
3      public String orderScreening(int quantity, boolean credit, int inventory) {
4          String output = null;
5          if ((quantity <= 1000) && credit)
6              if (quantity <= inventory)
7                  output = "Accept";
8              else
9                  output = "Defer";
10         else
11             output = "Reject";
12         return output;
13     }
14 }
```

Figure 1

Task 1

Based on the source code (*as shown in* Figure 1), construct the Control Flow Graph of the program.



Task 2

From the Control Flow Graph constructed in Task 1,

1) Identify the number of paths in the control flow graph using the Regular Expression approach (Show your step-by-step process)
2) Identify all the paths in the control flow graph using the Regular Expression approach (Show your step-by-step process).

Identify the Number of Paths: $1 \cdot (2 \cdot (4 + 5) + 3) \cdot 6$ $1 \cdot (1 \cdot (1+1) +1) \cdot 1 = 3$ (paths)
Identify All Paths: Path1: $1 \cdot 2 \cdot 4 \cdot 6$ Path2: $1 \cdot 2 \cdot 5 \cdot 6$ Path3: $1 \cdot 3 \cdot 6$

Task 3

Based on the paths identified in Task 2 and the program specification given at the beginning of the Problem 1, identify test cases and generate test data for the path coverage test.

Test Case	Nodes
P-1	1, 2, 4, 6
P-2	1, 2, 5, 6
P-3	1, 3, 6

Test ID	Test Cases Covered	Input			Exp. Output
		quantity	*credit*	*inventory*	*Return value*
T7.1	P-1	1000	true	1000	Accept
T7.2	P-2	1000	true	999	Defer
T7.3	P-3	1000	false	1000	Reject

Task 4

Based on the specification given above, write your testing code in JUnit 5 to test the source code of the program provided on Moodle ("***Lab7_Program1.java***"). Make sure your test code is named as "***Lab7_Task1.java***".

Program 2

Lab7_Program2 is a very special program that was developed to detect whether a string is an *odd string* or an *even string*. In this special program, a string is considered to be an odd string or even string based on the following rules:

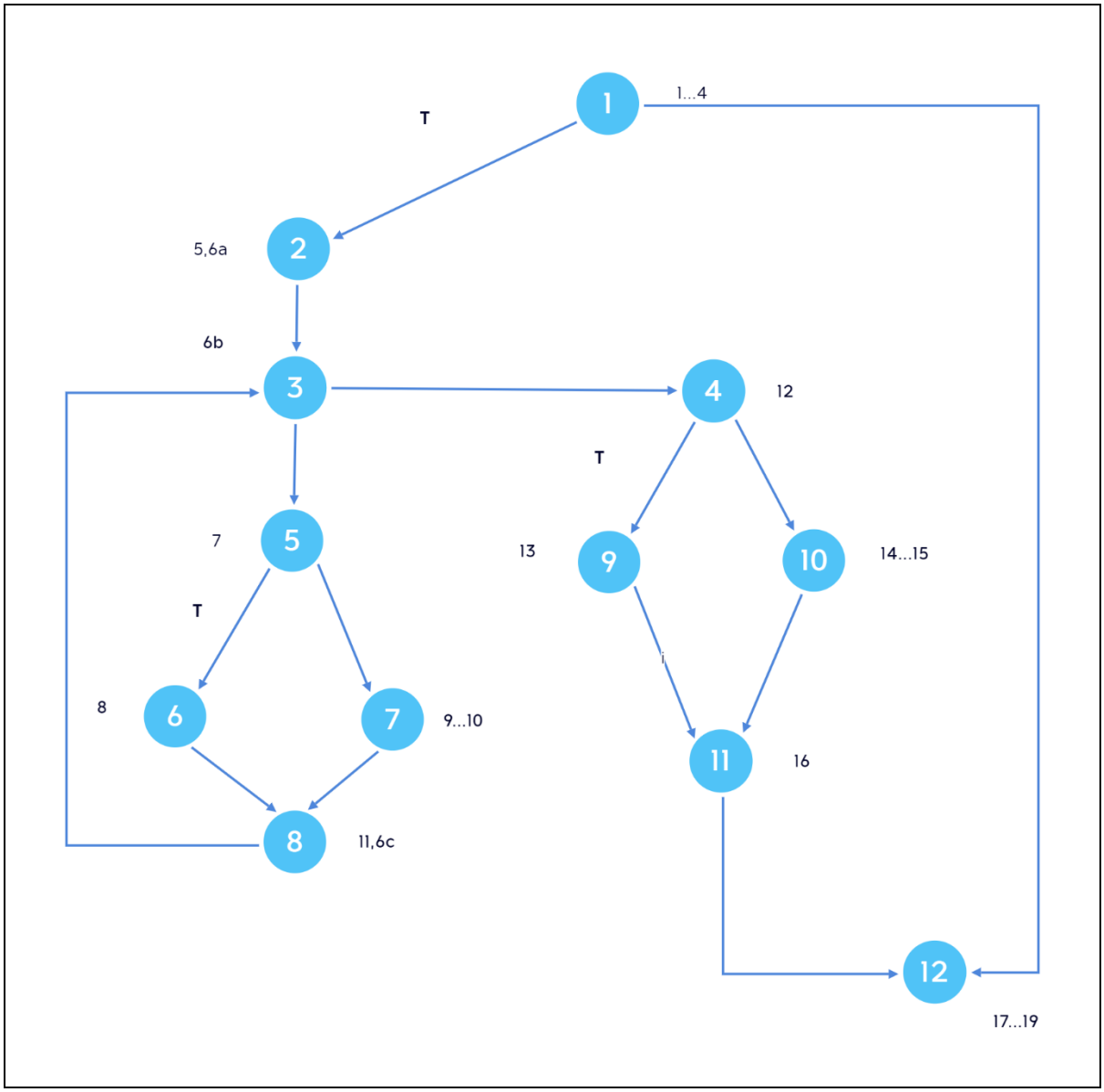
1. A string is a continuous series of characters. If the ASCII code value of a character is an even number, the character is considered as an even character, otherwise an odd character (see Appendix 1 for the ASCII code values).
2. If the number of odd characters is greater than the number of even characters, the program output "OddString", otherwise "EvenString".

```
1  public class Lab7_Program2 {
2      public String detectOddString(String str) {
3          String output = "Invalid";
4          if (str.length() > 0) {
5              int oddCounter = 0, evenCounter = 0;
6              for (int i = 0; i < str.length(); i++) {
7                  if (str.charAt(i) % 2 != 0)
8                      oddCounter++;
9                  else
10                     evenCounter++;
11             }
12             if (oddCounter > evenCounter)
13                 output = "OddString";
14             else
15                 output = "EvenString";
16         }
17         return output;
18     }
19 }
```

Figure 2

Task 1

Based on the source code (*as shown in* Figure 2), construct the Control Flow Graph of the program.



Task 2

From the Control Flow Graph constructed in Task 1,

1) Identify the number of paths in the control flow graph using the Regular Expression approach (Show your step-by-step process)
2) Identify all the paths in the control flow graph using the Regular Expression approach (Show your step-by-step process).

Identify the Number of Paths:
1 · (2 · (3 · 5 · (6 + 7) · 8) · 3)* 4 · (9 + 10) · 11 + 0) · 12 1 · (2 · (3 · 5 · (6 + 7) · 8) · 3 + 0) · 4 · (9 + 10) · 11 + 0) · 12 1 · (1 · (1 · 1 · (1 + 1) · 1) · 1 + 1) · (1 · (1 + 1) · 1 + 1) · 1 = 7 (paths)
Identify all paths: Path1: 1 · 2 · 3 · 4 · 9 · 11 · 12 Path2: 1 · 2 · 3 · 4 · 10 · 11 · 12 Path3: 1 · 2 · 3 · 5 · 6 · 8 · 3 · 4 · 9 · 11 · 12 Path4: 1 · 2 · 3 · 5 · 6 · 8 · 3 · 4 · 9 · 11 · 12 Path5: 1 · 2 · 3 · 5 · 7 · 8 · 3 · 4 · 9 · 11 · 12 Path6: 1 · 2 · 3 · 5 · 7 · 8 · 3 · 4 · 10 · 11 · 12 Path7: 1 · 12

Appendix 1: ASCII Code Table (Partial)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		