CMR
INSTITUTE OF
TECHNOLOGY

USN	1	C								

Internal Assessment Test 2 – November 2019

Sub:	Software Testing							Code:	18MCA351
Date:	19-11-19	Duration:	90 mins	Max Marks:	50	Sem:	III A Regular	Branch:	MCA

Note: Answer any full 5 questions. All questions carry equal marks. Total marks: 50

		Marks	OBE	
			CO	RBT
	Part I			
1.	a. Write algorithm for triangle problem b. Write EC test cases for Commission problem	[5+5]	CO1	L4,L1
	(OR)			
2.	Represent Next Date problem in a decision table and generate test cases from that	[5+5]	CO1	L1, L3
	Part II			
3.	Explain with appropriate diagrams the generation of test cases using EC technique for a function of two variables	[10]	CO1	L3, L2
	(OR)			
4.	a. Write test cases for commission problem using BVA. b. Write short note on SATM.	[5+5]	CO2	L2, L1
	Part III			
5.	a. What is path testing? Explain DU path testing technique. b. Find out DU path for variable 'total locks' in Commission problem.	[5+5]	CO1	L2,L4
	(OR)			
6.	a. What is DD path in code based testing? Illustrate with example. b. What is Slice-based testing? Compare it with DU path testing.	[5+5]	CO2	L2, L3
	Part IV			
7	Explain the Next Date problem and write the pseudo code.	[5+5]	CO2	L4, L1
	(OR)			
8	Explain with appropriate diagrams the generation of test cases using BVA technique for a function of two variables	[5+5]	CO2	L4, L4
	Part V			
9	Write pseudo code for Commission problem	[5+5]	CO2	L4, L4
	(OR)			
10	a. Write EC test cases for triangle problem. b. Write BV test cases for Next Date Problem	[5+5]	CO2	L2, L3

1. a.

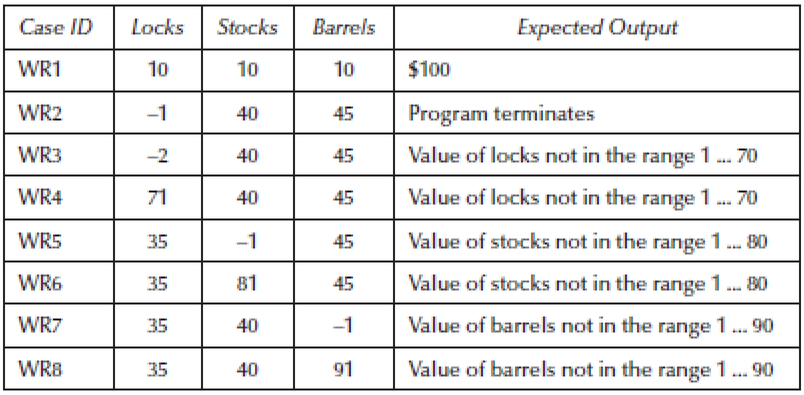
```
 Dim a, b, c As Integer
Dim c1, c2, c3, IsATriangle As Boolean
 'Step 1: Get Input
Do
Output( "Enter 3 integers which are sides of a triangle" )
Input(a, b, c)
c1 = (1 ≤ a) AND (a ≤ 300)
c2 = (1 ≤ b) AND (b ≤ 300)
c3 = (1 ≤ c) AND (c ≤ 300)
If NOT(c1)
Then Output( "Value of a is not in the range of permitted values" )
EndIf
If NOT(c2)
Then Output( "Value of b is not in the range of permitted values" )
EndIf
If NOT(c3)
ThenOutput( "Value of c is not in the range of permitted values" )
```

```
EndIf
Until c1 AND c2 AND c3
Output( “Side A is”, a)
Output( “Side B is”, b)
Output( “Side C is”, c)
 ‘Step 2: Is A Triangle?
If (a < b + c) AND (b < a + c) AND (c < a + b)
Then IsATriangle = True
Else IsATriangle = False
EndIf
 ‘Step 3: Determine Triangle Type
If IsATriangle
Then If (a = b) AND (b = c)
Then Output ( “Equilateral” )
Else If (a ≠ b) AND (a ≠ c) AND (b ≠ c)
Then Output ( “Scalene” )
Else Output ( “Isosceles” )
EndIf
EndIf
Else Output( “Not a Triangle” )
EndIf
End triangle3
```

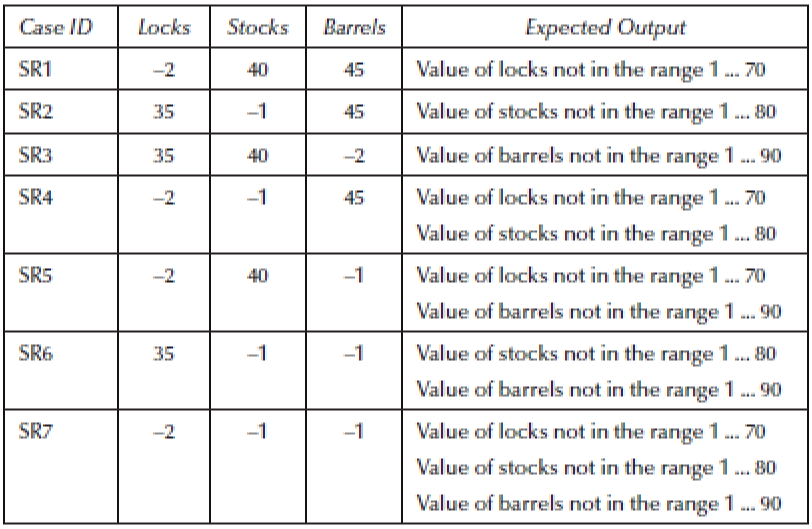
b.

The valid classes of the input variables are
L1 = {locks: 1 ≤ locks ≤ 70}
L2 = {locks = -1} (occurs if locks = -1 is used to control input iteration)
S1 = {stocks: 1 ≤ stocks ≤ 80}
B1 = {barrels: 1 ≤ barrels ≤ 90}
The corresponding invalid classes of the input variables are
L3 = {locks: locks = 0 OR locks < -1}
L4 = {locks: locks > 70}
S2 = {stocks: stocks < 1}
S3 = {stocks: stocks > 80}
B2 = {barrels: barrels < 1}
B3 = {barrels: barrels > 90}

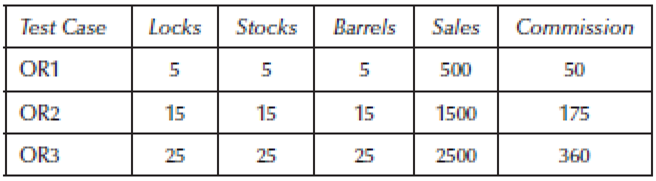
Test Cases:



Case ID	*Locks*	*Stocks*	*Barrels*	*Expected Output*
WR1	10	10	10	$100
WR2	–1	40	45	Program terminates
WR3	–2	40	45	Value of locks not in the range 1 ... 70
WR4	71	40	45	Value of locks not in the range 1 ... 70
WR5	35	–1	45	Value of stocks not in the range 1 ... 80
WR6	35	81	45	Value of stocks not in the range 1 ... 80
WR7	35	40	–1	Value of barrels not in the range 1 ... 90
WR8	35	40	91	Value of barrels not in the range 1 ... 90



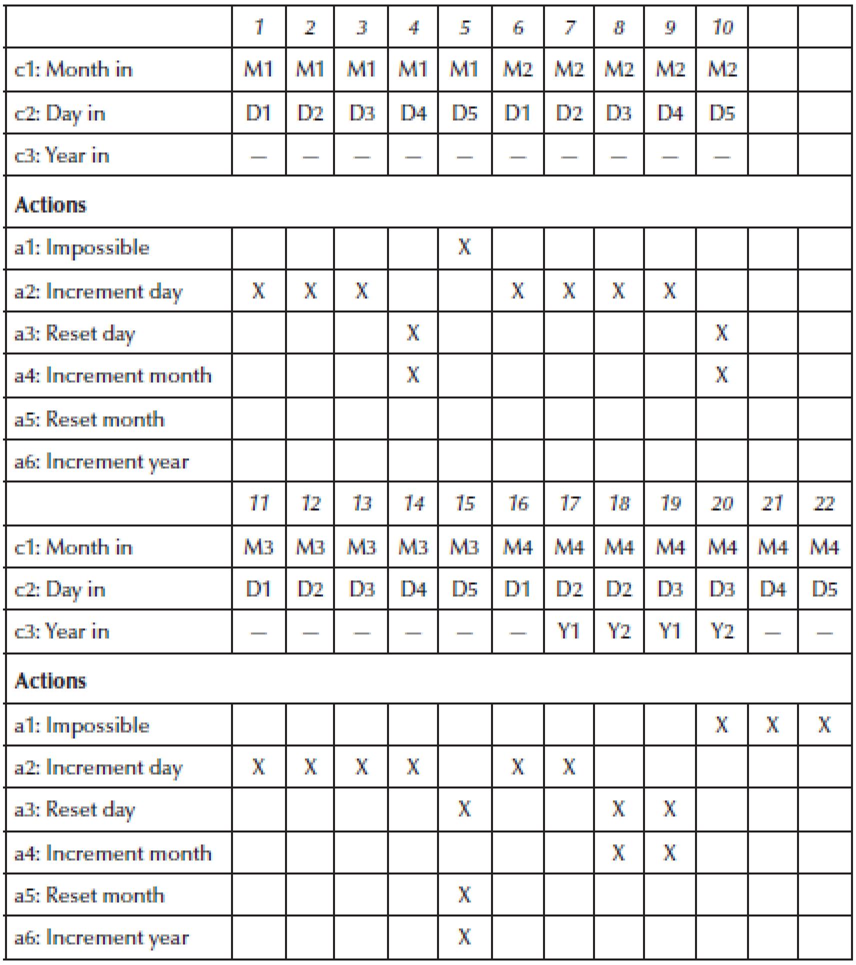
Case ID	*Locks*	*Stocks*	*Barrels*	*Expected Output*
SR1	–2	40	45	Value of locks not in the range 1 ... 70
SR2	35	–1	45	Value of stocks not in the range 1 ... 80
SR3	35	40	–2	Value of barrels not in the range 1 ... 90
SR4	–2	–1	45	Value of locks not in the range 1 ... 70 Value of stocks not in the range 1 ... 80
SR5	–2	40	–1	Value of locks not in the range 1 ... 70 Value of barrels not in the range 1 ... 90
SR6	35	–1	–1	Value of stocks not in the range 1 ... 80 Value of barrels not in the range 1 ... 90
SR7	–2	–1	–1	Value of locks not in the range 1 ... 70 Value of stocks not in the range 1 ... 80 Value of barrels not in the range 1 ... 90



Test Case	*Locks*	*Stocks*	*Barrels*	*Sales*	*Commission*
OR1	5	5	5	500	50
OR2	15	15	15	1500	175
OR3	25	25	25	2500	360

2.

Table 7.14 Decision Table for NextDate Function



	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*		
c1: Month in	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2		
c2: Day in	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5		
c3: Year in	—	—	—	—	—	—	—	—	—	—		
Actions												
a1: Impossible					X							
a2: Increment day	X	X	X			X	X	X	X			
a3: Reset day				X						X		
a4: Increment month				X						X		
a5: Reset month												
a6: Increment year												
	11	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
c1: Month in	M3	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4
c2: Day in	D1	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
c3: Year in	—	—	—	—	—	—	Y1	Y2	Y1	Y2	—	—
Actions												
a1: Impossible										X	X	X
a2: Increment day	X	X	X	X		X	X					
a3: Reset day					X			X	X			
a4: Increment month								X	X			
a5: Reset month					X							
a6: Increment year					X							

3.

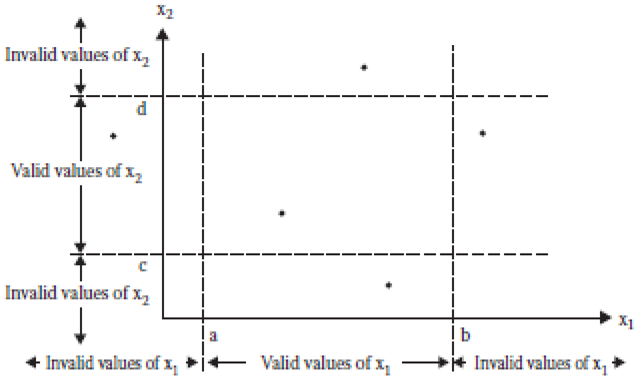


Figure 6.1 Traditional equivalence class test cases.

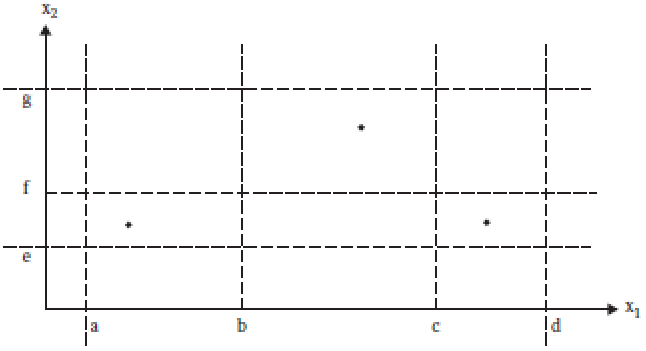


Figure 6.2 Weak normal equivalence class test cases.

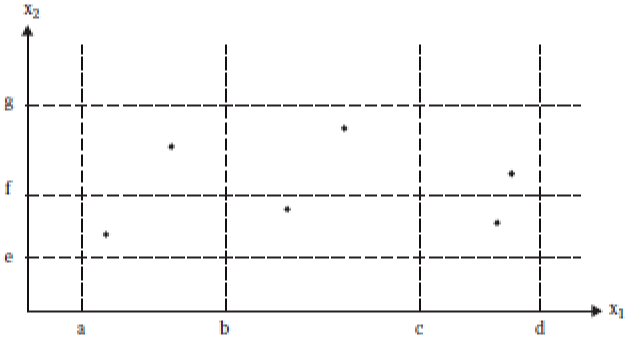


Figure 6.3 Strong normal equivalence class test cases.

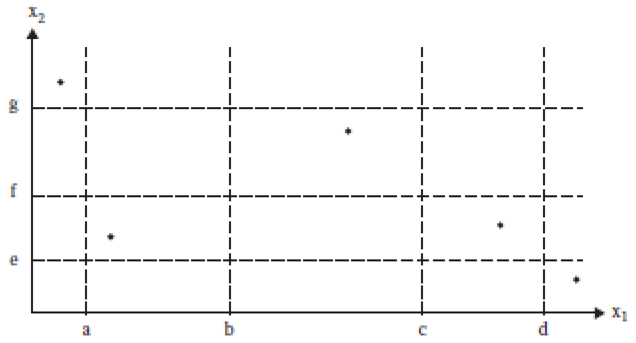


Figure 6.4 Weak robust equivalence class test cases.

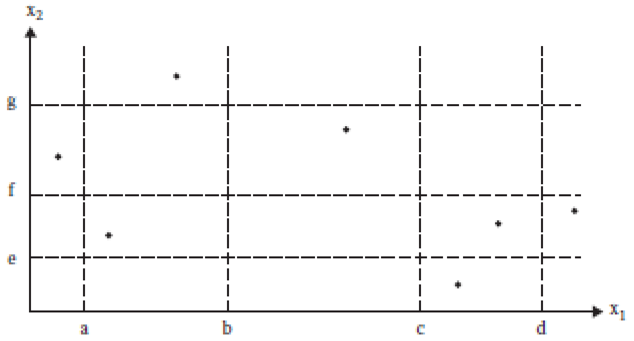


Figure 6.5 Revised weak robust equivalence class test cases.

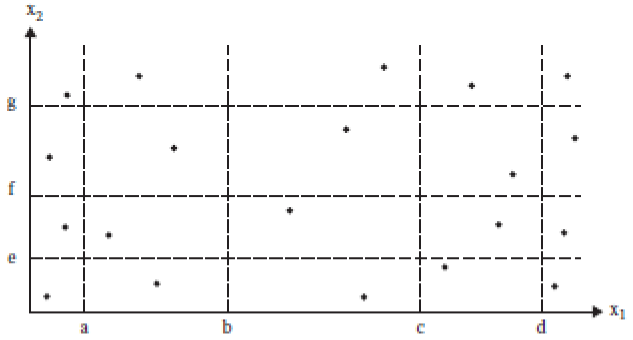
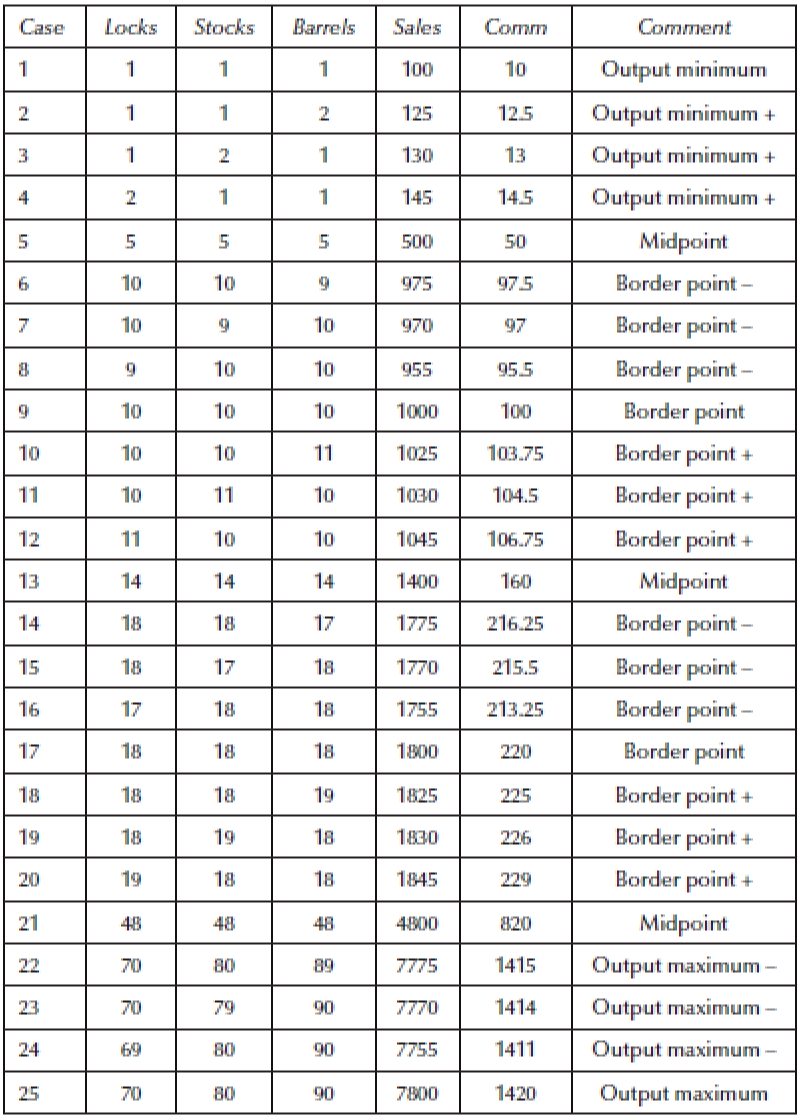


Figure 6.6 Strong robust equivalence class test cases.

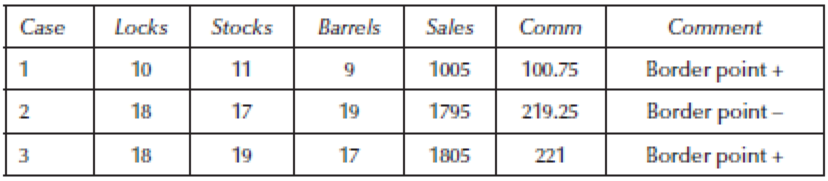
4. a.

Table 5.4 Output Boundary Value Analysis Test Cases



Case	Locks	Stocks	Barrels	Sales	Comm	Comment
1	1	1	1	100	10	Output minimum
2	1	1	2	125	12.5	Output minimum +
3	1	2	1	130	13	Output minimum +
4	2	1	1	145	14.5	Output minimum +
5	5	5	5	500	50	Midpoint
6	10	10	9	975	97.5	Border point –
7	10	9	10	970	97	Border point –
8	9	10	10	955	95.5	Border point –
9	10	10	10	1000	100	Border point
10	10	10	11	1025	103.75	Border point +
11	10	11	10	1030	104.5	Border point +
12	11	10	10	1045	106.75	Border point +
13	14	14	14	1400	160	Midpoint
14	18	18	17	1775	216.25	Border point –
15	18	17	18	1770	215.5	Border point –
16	17	18	18	1755	213.25	Border point –
17	18	18	18	1800	220	Border point
18	18	18	19	1825	225	Border point +
19	18	19	18	1830	226	Border point +
20	19	18	18	1845	229	Border point +
21	48	48	48	4800	820	Midpoint
22	70	80	89	7775	1415	Output maximum –
23	70	79	90	7770	1414	Output maximum –
24	69	80	90	7755	1411	Output maximum –
25	70	80	90	7800	1420	Output maximum

Table 5.5 Output Special Value Test Cases



Case	Locks	Stocks	Barrels	Sales	Comm	Comment
1	10	11	9	1005	100.75	Border point +
2	18	17	19	1795	219.25	Border point –
3	18	19	17	1805	221	Border point +

4. b.

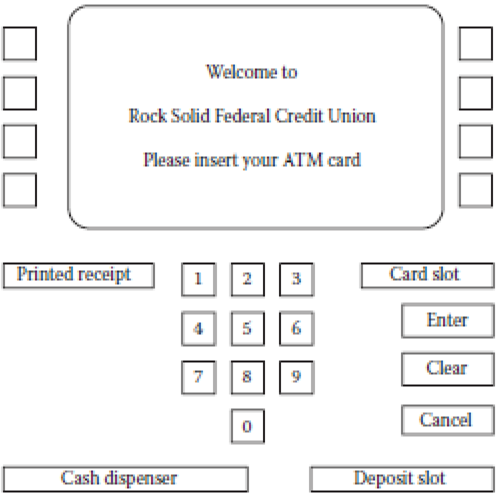
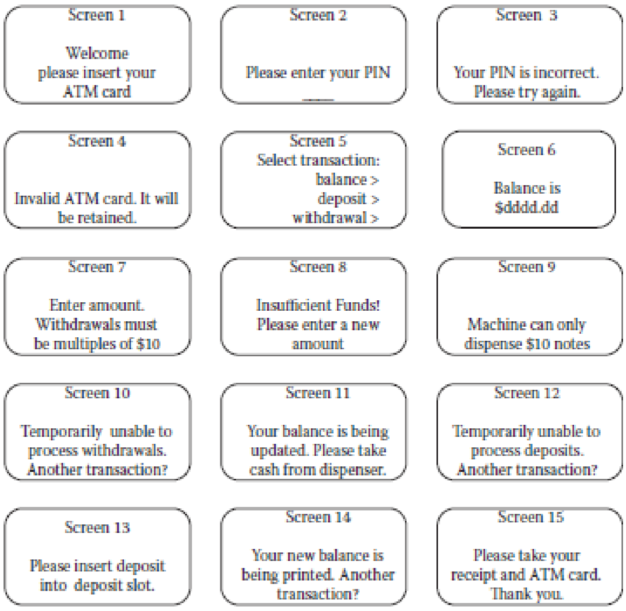


Figure 2.3 SATM terminal.



Screen 1

Welcome please insert your ATM card

Screen 2

Please enter your PIN ____

Screen 3

Your PIN is incorrect. Please try again.

Screen 4

Invalid ATM card. It will be retained.

Screen 5

Select transaction:
balance >
deposit >
withdrawal >

Screen 6

Balance is $dddd.dd

Screen 7

Enter amount. Withdrawals must be multiples of $10

Screen 8

Insufficient Funds! Please enter a new amount

Screen 9

Machine can only dispense $10 notes

Screen 10

Temporarily unable to process withdrawals. Another transaction?

Screen 11

Your balance is being updated. Please take cash from dispenser.

Screen 12

Temporarily unable to process deposits. Another transaction?

Screen 13

Please insert deposit into deposit slot.

Screen 14

Your new balance is being printed. Another transaction?

Screen 15

Please take your receipt and ATM card. Thank you.

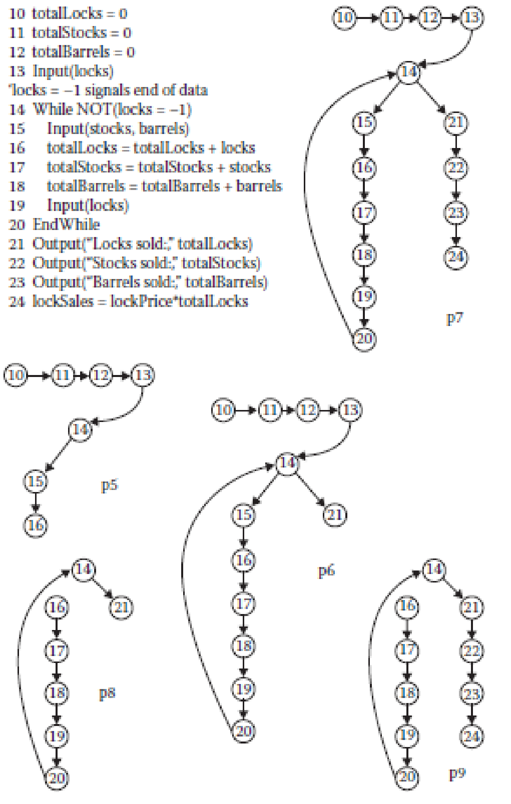
Figure 2.4 SATM screens.

5. a.

DU Path

The following definitions refer to a program *P* that has a program graph *G*(*P*) and a set of program variables *V*. *G*(*P*) has a single-entry node and a single-exit node. We also disallow edges from a node to itself. Paths, subpaths, and cycles are as they were in Chapter 4. The set of all paths in *P* is PATHS(*P*). Node *n* ∈ *G*(*P*) is a *defining node* of the variable *v* ∈ *V*, written as DEF(*v*, *n*), if and only if the value of variable *v* is defined as the statement fragment corresponding to node *n*. Node *n* ∈ *G*(*P*) is a *usage node* of the variable *v* ∈ *V*, written as USE(*v*, *n*), if and only if the value of the variable *v* is used as the statement fragment corresponding to node *n*. A usage node USE(*v*, *n*) is a *predicate use* (denoted as P-use) if and only if the statement *n* is a predicate statement; otherwise, USE(*v*, *n*) is a computation use (denoted C-use). A *definition/use path* with respect to a variable *v* (denoted du-path) is a path in PATHS(*P*) such that, for some *v* ∈ *V*, there are define and usage nodes DEF(*v*, *m*) and USE(*v*, *n*) such that *m* and *n* are the initial and final nodes of the path. A *definition-clear path* with respect to a variable *v* (denoted dc-path) is a definition/use path in PATHS(*P*) with initial and final nodes DEF(*v*, *m*) and USE(*v*, *n*) such that no other node in the path is a defining node of *v*.

b.



```
10 totalLocks = 0
11 totalStocks = 0
12 totalBarrels = 0
13 Input(locks)
'locks = –1 signals end of data
14 While NOT(locks = –1)
15    Input(stocks, barrels)
16    totalLocks = totalLocks + locks
17    totalStocks = totalStocks + stocks
18    totalBarrels = totalBarrels + barrels
19    Input(locks)
20 EndWhile
21 Output("Locks sold:," totalLocks)
22 Output("Stocks sold:," totalStocks)
23 Output("Barrels sold:," totalBarrels)
24 lockSales = lockPrice*totalLocks
```

Figure 9.4 Du-paths for totalLocks.

6.a.

A *DD-path* is a sequence of nodes in a program graph such that
Case 1: It consists of a single node with indeg = 0.
Case 2: It consists of a single node with outdeg = 0.
Case 3: It consists of a single node with indeg ≥ 2 or outdeg ≥ 2.
Case 4: It consists of a single node with indeg = 1 and outdeg = 1.
Case 5: It is a maximal chain of length ≥ 1.

Figure 8.2 Nodes	DD-Path	Case of definition
4	First	1
5-8	A	5
9	B	3
10	C	4
11	D	4
12	E	3
13	F	3
14	H	3
15	I	4
16	J	3
17	K	4
18	L	4
19	M	3
20	N	3
21	G	4
22	O	3
23	Last	2

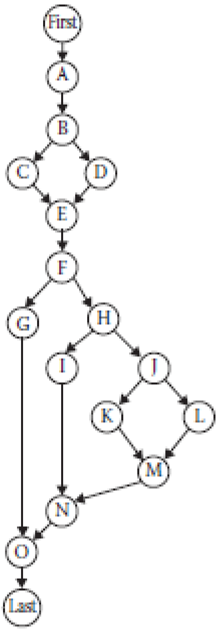
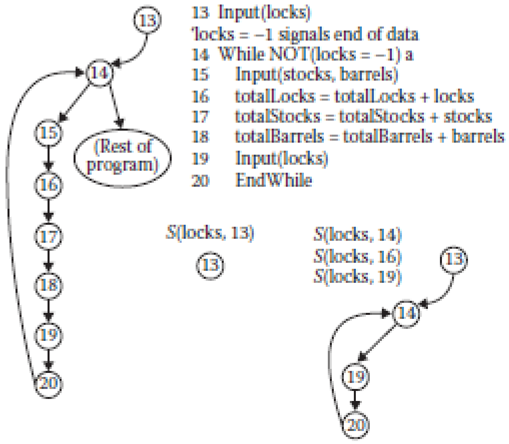


Figure 8.5 DD-path graph for triangle program.

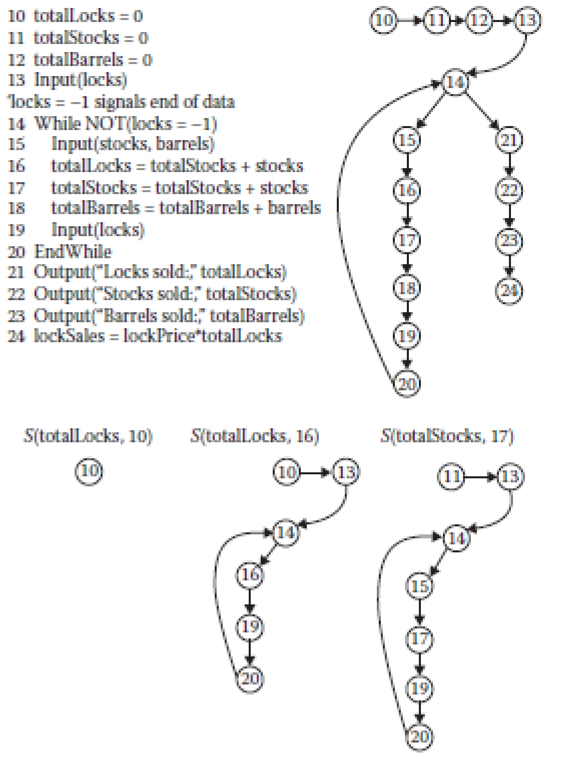
b. Given a program *P* and a set *V* of variables in *P*, a *slice on the variable set V at statement n*, written *S*(*V*, *n*), is the set of all statement fragments in *P* that contribute to the values of variables in *V* at node *n*.

Given a program *P* and a program graph *G*(*P*) in which statements and statement fragments are numbered, and a set *V* of variables in *P*, *the static, backward slice on the variable set V at statement fragment n*, written *S*(*V*, *n*), is the set of node numbers of all statement fragments in *P* that contribute to the values of variables in *V* at statement fragment *n*.



```
13 Input(locks)
'locks = −1 signals end of data
14 While NOT(locks = −1) a
15     Input(stocks, barrels)
16     totalLocks = totalLocks + locks
17     totalStocks = totalStocks + stocks
18     totalBarrels = totalBarrels + barrels
19     Input(locks)
20     EndWhile
```

Figure 9.6 Selected slices on locks.



```
10 totalLocks = 0
11 totalStocks = 0
12 totalBarrels = 0
13 Input(locks)
'locks = –1 signals end of data
14 While NOT(locks = –1)
15    Input(stocks, barrels)
16    totalLocks = totalStocks + stocks
17    totalStocks = totalStocks + stocks
18    totalBarrels = totalBarrels + barrels
19    Input(locks)
20 EndWhile
21 Output("Locks sold:," totalLocks)
22 Output("Stocks sold:," totalStocks)
23 Output("Barrels sold:," totalBarrels)
24 lockSales = lockPrice*totalLocks
```

igure 9.7 Selected slices in a loop.

7.

```
Program NextDate1  ‘Simple version
Dim tomorrowDay,tomorrowMonth,tomorrowYear As Integer
Dim day,month,year As Integer
Output ( “Enter today’ s date in the form MM DD YYYY” )
Input (month, day, year)
Case month Of
Case 1： month Is 1,3,5,7,8, Or 10：  ‘31 day months (except Dec.)
If day < 31
Then tomorrowDay = day + 1
Else
tomorrowDay = 1
tomorrowMonth = month + 1
EndIf
Case 2： month Is 4,6,9, Or 11  ‘30 day months
If day < 30
Then tomorrowDay = day + 1
Else
tomorrowDay = 1
tomorrowMonth = month + 1
EndIf
Case 3： month Is 12：  ‘December
If day < 31
Then tomorrowDay = day + 1
Else
tomorrowDay = 1
tomorrowMonth = 1
If year = 2012
Then Output ( “2012 is over” )
Else tomorrow.year = year + 1
EndIf
Case 4： month is 2：  ‘February
If day < 28
Then tomorrowDay = day + 1
Else
If day = 28
Then If ((year is a leap year)
Then tomorrowDay = 29  ‘leap year
Else  ‘not a leap year
```

```
tomorrowDay = 1
tomorrowMonth = 3
EndIf
Else If day = 29
Then If ((year is a leap year)
Then tomorrowDay = 1
tomorrowMonth = 3
Else  ‘not a leap year
Output( “Cannot have Feb.” , day)
EndIf
EndIf
EndIf
EndIf
EndCase
Output ( “Tomorrow’ s date is” , tomorrowMonth, tomorrowDay, tomorrowYear)
End NextDate
Program NextDate2 Improved version
 ‘
Dim tomorrowDay,tomorrowMonth,tomorrowYear As Integer
Dim day,month,year As Integer
Dim c1, c2, c3 As Boolean
 ‘
Do
Output ( “Enter today’ s date in the form MM DD YYYY” )
Input (month, day, year)
c1 = (1 ≤ day) AND (day ≤ 31)
c2 = (1 ≤ month) AND (month ≤ 12)
c3 = (1812 ≤ year) AND (year ≤ 2012)
If NOT(c1)
Then Output( “Value of day not in the range 1..31” )
EndIf
If NOT(c2)
Then Output( “Value of month not in the range 1..12” )
EndIf
If NOT(c3)
Then Output( “Value of year not in the range 1812..2012” )
EndIf
Until c1 AND c2 AND c2
Case month Of
Case 1: month Is 1,3,5,7,8, Or 10:  ‘31 day months (except Dec.)
If day < 31
Then tomorrowDay = day + 1
Else
tomorrowDay = 1
tomorrowMonth = month + 1
EndIf
Case 2: month Is 4,6,9, Or 11  ‘30 day months
If day < 30
Then tomorrowDay = day + 1
Else
If day = 30
Then tomorrowDay = 1
tomorrowMonth = month + 1
Else Output( “Invalid Input Date” )
EndIf
EndIf
Case 3: month Is 12:  ‘December
If day < 31
Then tomorrowDay = day + 1
Else
tomorrowDay = 1
tomorrowMonth = 1
If year = 2012
Then Output ( “Invalid Input Date” )
Else tomorrow.year = year + 1
EndIf
EndIf
```

```
Case 4: month is 2:  ‘February
If day < 28
Then tomorrowDay = day + 1
Else
If day = 28
Then
If (year is a leap year)
Then tomorrowDay = 29  ‘leap day
Else  ‘not a leap year
tomorrowDay = 1
tomorrowMonth = 3
EndIf
Else
If day = 29
Then
If (year is a leap year)
Then tomorrowDay = 1
tomorrowMonth = 3
Else
If day > 29
Then Output( “Invalid Input Date” )
EndIf
EndIf
EndIf
EndIf
EndIf
EndCase
Output ( “Tomorrow’ s date is” , tomorrowMonth, tomorrowDay, tomorrowYear)
 ‘
End NextDate2
```

8.

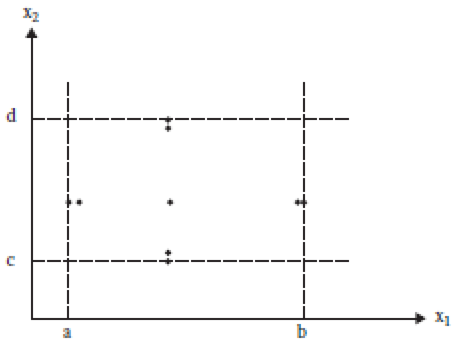


Figure 5.2 Boundary value analysis test cases for a function of two variables.

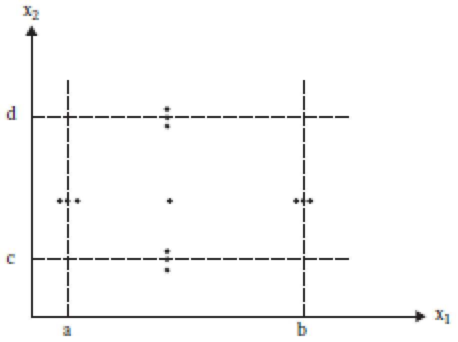


Figure 5.3 Robustness test cases for a function of two variables.

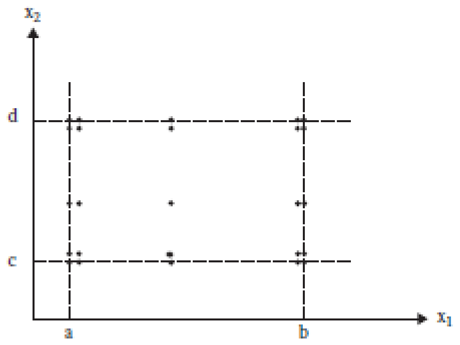


Figure 5.4 Worst-case test cases for a function of two variables.

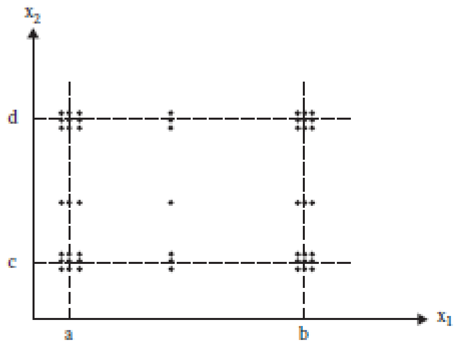
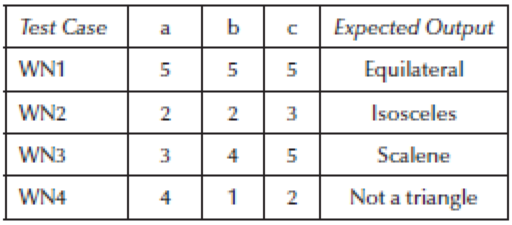


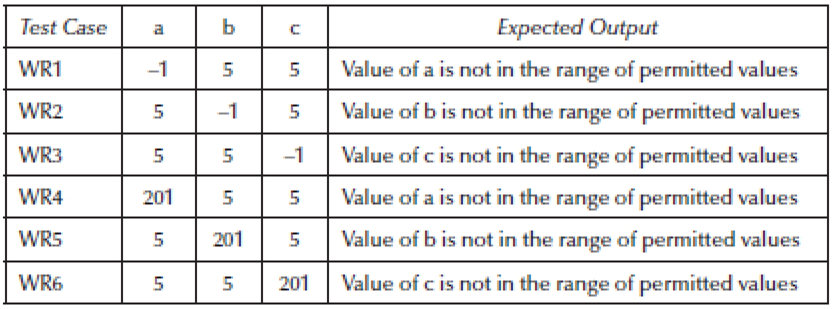
Figure 5.5 Robust worst-case test cases for a function of two variables.

```
9. Program Commission (INPUT,OUTPUT)
 ‘
Dim locks, stocks, barrels As Integer
Dim lockPrice, stockPrice, barrelPrice As Real
Dim totalLocks,totalStocks,totalBarrels As Integer
Dim lockSales, stockSales, barrelSales As Real
Dim sales,commission : REAL
 ‘
lockPrice = 45.0
stockPrice = 30.0
barrelPrice = 25.0
totalLocks = 0
totalStocks = 0
totalBarrels = 0
 ‘
Input(locks)
While NOT(locks = -1)  ‘Input device uses -1 to indicate end of data
Input(stocks, barrels)
totalLocks = totalLocks + locks
totalStocks = totalStocks + stocks
totalBarrels = totalBarrels + barrels
Input(locks)
EndWhile
 ‘
Output( “Locks sold:” , totalLocks)
Output( “Stocks sold:” , totalStocks)
Output( “Barrels sold:” , totalBarrels)
 ‘
lockSales = lockPrice * totalLocks
stockSales = stockPrice * totalStocks
barrelSales = barrelPrice * totalBarrels
sales = lockSales + stockSales + barrelSales
Output( “Total sales:” , sales)
 ‘
If (sales > 1800.0)
Then
commission = 0.10 * 1000.0
commission = commission + 0.15 * 800.0
commission = commission + 0.20 * (sales-1800.0)
Else If (sales > 1000.0)
Then
commission = 0.10 * 1000.0
commission = commission + 0.15*(sales-1000.0)
Else commission = 0.10 * sales
EndIf
EndIf
Output( “Commission is $” ,commission)
End Commission
```

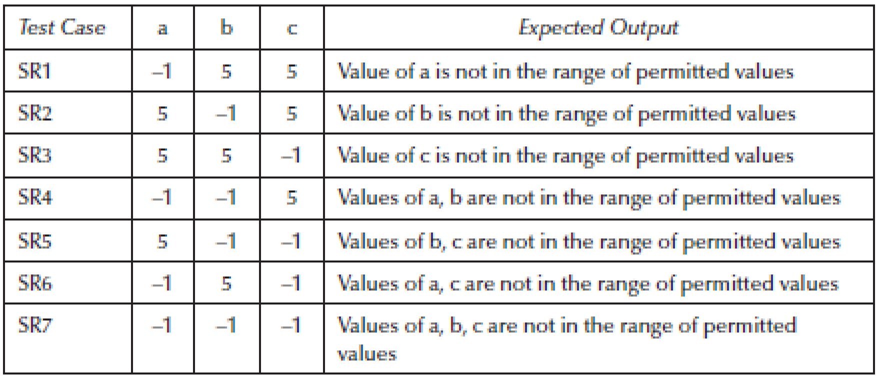
10.a.



Test Case	a	b	c	*Expected Output*
WN1	5	5	5	Equilateral
WN2	2	2	3	Isosceles
WN3	3	4	5	Scalene
WN4	4	1	2	Not a triangle



Test Case	a	b	c	*Expected Output*
WR1	–1	5	5	Value of a is not in the range of permitted values
WR2	5	–1	5	Value of b is not in the range of permitted values
WR3	5	5	–1	Value of c is not in the range of permitted values
WR4	201	5	5	Value of a is not in the range of permitted values
WR5	5	201	5	Value of b is not in the range of permitted values
WR6	5	5	201	Value of c is not in the range of permitted values



Test Case	a	b	c	*Expected Output*
SR1	–1	5	5	Value of a is not in the range of permitted values
SR2	5	–1	5	Value of b is not in the range of permitted values
SR3	5	5	–1	Value of c is not in the range of permitted values
SR4	–1	–1	5	Values of a, b are not in the range of permitted values
SR5	5	–1	–1	Values of b, c are not in the range of permitted values
SR6	–1	5	–1	Values of a, c are not in the range of permitted values
SR7	–1	–1	–1	Values of a, b, c are not in the range of permitted values

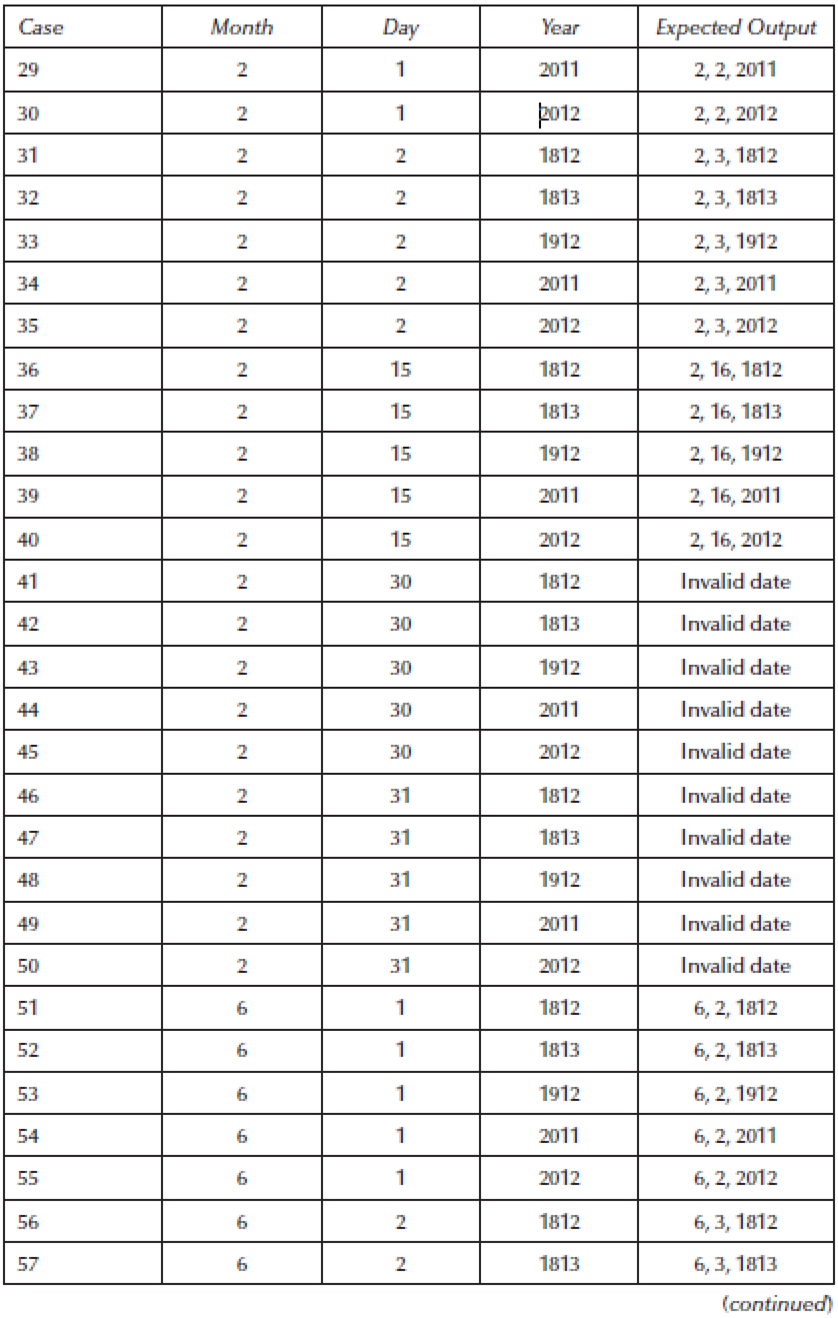
b.

Table 5.1 Normal Boundary Value Test Cases

Case	a	b	c	*Expected Output*
1	100	100	1	Isosceles
2	100	100	2	Isosceles
3	100	100	100	Equilateral
4	100	100	199	Isosceles
5	100	100	200	Not a triangle
6	100	1	100	Isosceles
7	100	2	100	Isosceles
8	100	100	100	Equilateral
9	100	199	100	Isosceles
10	100	200	100	Not a triangle
11	1	100	100	Isosceles
12	2	100	100	Isosceles
13	100	100	100	Equilateral
14	199	100	100	Isosceles
15	200	100	100	Not a triangle

Case	*Month*	*Day*	*Year*	*Expected Output*
1	1	1	1812	1, 2, 1812
2	1	1	1813	1, 2, 1813
3	1	1	1912	1, 2, 1912
4	1	1	2011	1, 2, 2011
5	1	1	2012	1, 2, 2012
6	1	2	1812	1, 3, 1812
7	1	2	1813	1, 3, 1813
8	1	2	1912	1, 3, 1912
9	1	2	2011	1, 3, 2011
10	1	2	2012	1, 3, 2012
11	1	15	1812	1, 16, 1812
12	1	15	1813	1, 16, 1813
13	1	15	1912	1, 16, 1912
14	1	15	2011	1, 16, 2011
15	1	15	2012	1, 16, 2012
16	1	30	1812	1, 31, 1812
17	1	30	1813	1, 31, 1813
18	1	30	1912	1, 31, 1912
19	1	30	2011	1, 31, 2011
20	1	30	2012	1, 31, 2012
21	1	31	1812	2, 1, 1812
22	1	31	1813	2, 1, 1813
23	1	31	1912	2, 1, 1912
24	1	31	2011	2, 1, 2011
25	1	31	2012	2, 1, 2012
26	2	1	1812	2, 2, 1812
27	2	1	1813	2, 2, 1813
28	2	1	1912	2, 2, 1912

(continued)



Case	Month	Day	Year	Expected Output
29	2	1	2011	2, 2, 2011
30	2	1	2012	2, 2, 2012
31	2	2	1812	2, 3, 1812
32	2	2	1813	2, 3, 1813
33	2	2	1912	2, 3, 1912
34	2	2	2011	2, 3, 2011
35	2	2	2012	2, 3, 2012
36	2	15	1812	2, 16, 1812
37	2	15	1813	2, 16, 1813
38	2	15	1912	2, 16, 1912
39	2	15	2011	2, 16, 2011
40	2	15	2012	2, 16, 2012
41	2	30	1812	Invalid date
42	2	30	1813	Invalid date
43	2	30	1912	Invalid date
44	2	30	2011	Invalid date
45	2	30	2012	Invalid date
46	2	31	1812	Invalid date
47	2	31	1813	Invalid date
48	2	31	1912	Invalid date
49	2	31	2011	Invalid date
50	2	31	2012	Invalid date
51	6	1	1812	6, 2, 1812
52	6	1	1813	6, 2, 1813
53	6	1	1912	6, 2, 1912
54	6	1	2011	6, 2, 2011
55	6	1	2012	6, 2, 2012
56	6	2	1812	6, 3, 1812
57	6	2	1813	6, 3, 1813

(continued)

Case	*Month*	*Day*	*Year*	*Expected Output*
58	6	2	1912	6, 3, 1912
59	6	2	2011	6, 3, 2011
60	6	2	2012	6, 3, 2012
61	6	15	1812	6, 16, 1812
62	6	15	1813	6, 16, 1813
63	6	15	1912	6, 16, 1912
64	6	15	2011	6, 16, 2011
65	6	15	2012	6, 16, 2012
66	6	30	1812	7, 1, 1812
67	6	30	1813	7, 1, 1813
68	6	30	1912	7, 1, 1912
69	6	30	2011	7, 1, 2011
70	6	30	2012	7, 1, 2012
71	6	31	1812	Invalid date
72	6	31	1813	Invalid date
73	6	31	1912	Invalid date
74	6	31	2011	Invalid date
75	6	31	2012	Invalid date
76	11	1	1812	11, 2, 1812
77	11	1	1813	11, 2, 1813
78	11	1	1912	11, 2, 1912
79	11	1	2011	11, 2, 2011
80	11	1	2012	11, 2, 2012
81	11	2	1812	11, 3, 1812
82	11	2	1813	11, 3, 1813
83	11	2	1912	11, 3, 1912
84	11	2	2011	11, 3, 2011
85	11	2	2012	11, 3, 2012
86	11	15	1812	11, 16, 1812

(*continued*)

Case	*Month*	*Day*	*Year*	*Expected Output*
87	11	15	1813	11, 16, 1813
88	11	15	1912	11, 16, 1912
89	11	15	2011	11, 16, 2011
90	11	15	2012	11, 16, 2012
91	11	30	1812	12, 1, 1812
92	11	30	1813	12, 1, 1813
93	11	30	1912	12, 1, 1912
94	11	30	2011	12, 1, 2011
95	11	30	2012	12, 1, 2012
96	11	31	1812	Invalid date
97	11	31	1813	Invalid date
98	11	31	1912	Invalid date
99	11	31	2011	Invalid date
100	11	31	2012	Invalid date
101	12	1	1812	12, 2, 1812
102	12	1	1813	12, 2, 1813
103	12	1	1912	12, 2, 1912
104	12	1	2011	12, 2, 2011
105	12	1	2012	12, 2, 2012
106	12	2	1812	12, 3, 1812
107	12	2	1813	12, 3, 1813
108	12	2	1912	12, 3, 1912
109	12	2	2011	12, 3, 2011
110	12	2	2012	12, 3, 2012
111	12	15	1812	12, 16, 1812
112	12	15	1813	12, 16, 1813
113	12	15	1912	12, 16, 1912
114	12	15	2011	12, 16, 2011
115	12	15	2012	12, 16, 2012
116	12	30	1812	12, 31, 1812
117	12	30	1813	12, 31, 1813
118	12	30	1912	12, 31, 1912
119	12	30	2011	12, 31, 2011
120	12	30	2012	12, 31, 2012
121	12	31	1812	1, 1, 1813
122	12	31	1813	1, 1, 1814
123	12	31	1912	1, 1, 1913
124	12	31	2011	1, 1, 2012
125	12	31	2012	1, 1, 2013

(continued)