Project Report

Software Testing and Analysis (CS 589)

Professor: Bogdan Korel

Report by:

Mohammed Shethwala

A20368337

Table of Contents

1. Introduction:

This project covers the two most important areas of software testing which are Model Based Testing and Code Based Testing. In this project, an EFSM model of a Vending Machine is used to derive the Transition-Pairing Test Cases and a VendingMachine class is used to perform the Multiple-Condition Testing. This project also covers Ghost/Default Transition Testing. A Test Driver and a Test Suite has also been prepared. The Test Driver will be used to test all the test cases derived from the three methods discussed above.

2. Model Based Testing:

From the EFSM model, I identified the transition pairs for all the states. After the pairs are identified, test cases were written covering all the transition pairs. There are some pairs which are non-executable and all such pairs are explained so as to why are they non-executable. I have also prepared a table which lists all the transition pairs and their corresponding test cases.

2.1. Identified Transition Pairs:

State	Incoming	Outgoing	Transition Pairs
Idle	T1,T2,T3,T4,T6, T8,T9,T10,T11, T12,T13,T14,T15	T2,T3,T4, T5,T6,T7	(T1,T2),(T1,T3),(T1,T4),(T1,T5),(T1,T6),(T1,T7), (T2,T2),(T2,T3),(T2,T4),(T2,T5),(T2,T6),(T2,T7), (T3,T2),(T3,T3),(T3,T4),(T3,T5),(T3,T6),(T3,T7), (T4,T2),(T4,T3),(T4,T4),(T4,T5),(T4,T6),(T4,T7), (T6,T2),(T6,T3),(T6,T4),(T6,T5),(T6,T6),(T6,T7), (T8,T2),(T8,T3),(T8,T4),(T8,T5),(T8,T6),(T8,T7), (T9,T2),(T9,T3),(T9,T4),(T9,T5),(T9,T6),(T9,T7), (T10,T2),(T10,T3),(T10,T4),(T10,T5),(T10,T6),(T10,T7), (T11,T2),(T11,T3),(T11,T4),(T11,T5),(T11,T6),(T11,T7), (T12,T2),(T12,T3),(T12,T4),(T12,T5),(T12,T6),(T12,T7), (T13,T2),(T13,T3),(T13,T4),(T13,T5),(T13,T6),(T13,T7), (T14,T2),(T14,T3),(T14,T4),(T14,T5),(T14,T6),(T14,T7), (T15,T2),(T15,T3),(T15,T4),(T15,T5),(T15,T6),(T15,T7)

Coins Inserted	T7,T19,T20, T21,T23	T10,T11,T12, T19,T20,T21, T22,T24,T25	(T7,T10),(T7,T11),(T7,T12),(T7,T19),(T7,T20), (T7,T21),(T7,T22),(T7,T24),(T7,T25) (T19,T10),(T19,T11),(T19,T12),(T19,T19),(T19,T20), (T19,T21),(T19,T22),(T19,T24),(T19,T25) (T20,T10),(T20,T11),(T20,T12),(T20,T19),(T20,T20), (T20,T21),(T20,T22),(T20,T24),(T20,T25) (T21,T10),(T21,T11),(T21,T12),(T21,T19),(T21,T20), (T21,T21),(T21,T22),(T21,T24),(T21,T25) (T23,T10),(T23,T11),(T23,T12),(T23,T19),(T23,T20), (T23,T21),(T23,T22),(T23,T24),(T23,T25)
Sugar	T16,T17, T18,T22	T13,T14,T15, T16,T17,T18, T23,T26,T27	(T16,T13),(T16,T14),(T16,T15),(T16,T16),(T16,T17), (T16,T18),(T16,T23),(T16,T26),(T16,T27) (T17,T13),(T17,T14),(T17,T15),(T17,T16),(T17,T17), (T17,T18),(T17,T23),(T17,T26),(T17,T27) (T18,T13),(T18,T14),(T18,T15),(T18,T16),(T18,T17), (T18,T18),(T18,T23),(T18,T26),(T18,T27) (T22,T13),(T22,T14),(T22,T15),(T22,T16),(T22,T17), (T22,T18),(T22,T23),(T22,T26),(T22,T27)
No Small Cups	T25,T27,T28	T9,T28	(T25,T9),(T25,T28),(T27,T9),(T27,T28), (T28,T9),(T28,T28)
No Large Cups	T24,T26,T29	T8,T29	(T24,T8),(T24,T29), (T26,T8),(T26,T29), (T29,T8),(T29,T29),

2.2. Test Cases For Transition Pairing Testing:

Here, the transition T1 which is the constructor vending_machine() will always be called and executed for all the tests. So, I have not included the vending_machine() constructor in the test cases, but I have included the transition T1 in every case so as to suggest that it is always executed first as the test case starts.

Test No:	Test#1
Test	set_price 25 insert_large_cups 5 insert_large_cups 5 insert_small_cups 5 insert_small_cups 5 insert_large_cups 5 set_price 50 insert_small_cups 5 set_price 50 set_price 75 coin coin set_price 100 insert_large_cups 5 coin insert_small_cups 5 coin coin coin small_cup small_cup sugar small_cup small_cup large_cup tea dispose
Transitions	T1,T4,T2,T2,T3,T3,T2,T4,T3,T4,T4,T6,T6,T4,T2,T6,T3,T7,T20,T20,T21,T21,T22,T17,T17,T18,T13,T5
Pairs Covered	(T1,T4),(T4,T2)(T2,T2),(T2,T3),(T3,T3),(T3,T2),(T2,T4),(T4,T3),(T3,T4),(T4,T4),(T4,T6),(T6,T6),(T6,T4),(T2,T6),(T6,T3),(T3,T7),(T7,T20),(T20,T20),(T20,T21),(T21,T21),(T21,T22),(T22,T17),(T17,T17),(T17,T18),(T18,T13),(T13,T5)

Test No:	Test#2
Test	set_price 25 coin cancel set_price 50 coin coin large_cup large_cup coin large_cup small_cup coin small_cup sugar coin coin small_cup small_cup large_cup large_cup small_cup coin sugar coin sugar sugar small_cup sugar sugar large_cup cancel insert_large_cups 1 dispose
Transitions	T1,T4,T7,T10,T4,T6,T7,T19,T19,T20,T19,T21,T20,T21,T22,T16,T16,T17,T17,T18,T18,T17,T16,T23,T20,T22,T23,T21,T22,T23,T19,T10,T2,T5
Pairs Covered	(T4,T7),(T7,T10),(T10,T4),(T6,T7),(T7,T19),(T19,T19),(T19,T20),(T20,T19),(T19,T21),(T21,T20),(T22,T16),(T16,T16),(T16,T17),(T18,T18),(T18,T17),(T17,T16),(T16,T23),(T23,T20),(T20,T22),(T22,T23),(T23,T21),(T23,T19),(T19,T10),(T10,T2),(T2,T5)

Test No:	Test#3
Test	set_price 50 coin insert_large_cups 1 coin large_cup tea coin coin insert_large_cups 1 insert_large_cups 1 dispose
Transitions	T1,T4,T6,T2,T7,T19,T24,T29,T29,T8,T2,T5
Pairs Covered	(T6,T2),(T2,T7),(T19,T24),(T24,T29),(T29,T29),(T29,T8),(T8,T2)

Test No:	Test#4

Test	set_price 25 insert_large_cups 1 coin small_cup large_cup sugar large_cup coin large_cup tea coin insert_large_cups 1 insert_small_cups 1 dispose
Transitions	T1,T4,T2,T7,T21,T19,T22,T18,T16,T18,T26,T29,T8,T3,T5
Pairs Covered	(T7,T21),(T21,T19),(T19,T22),(T22,T18),(T18,T16),(T16,T18),(T18,T26),(T26,T29), (T8,T3),(T3,T5)

Test No:	Test#5
Test	set_price 25 insert_large_cups 5 insert_small_cups 1 coin sugar small_cup tea coin coin insert_small_cups 1 insert_small_cups 1 dispose
Transitions	T1,T4,T2,T3,T7,T22,T17,T27,T28,T28,T9,T3,T5
Pairs Covered	(T7,T22),(T17,T27),(T27,T28),(T28,T28),(T28,T9),(T9,T3)

Test No:	Test#6
Test	set_price 25 insert_small_cups 1 coin small_cup tea insert_small_cups 1 insert_large_cups 1 coin small_cup sugar tea insert_small_cups 1 insert_small_cups 1 dispose
Transitions	T1,T4,T3,T7,T21,T25,T9,T2,T7,T21,T22,T27,T9,T3,T5
Pairs Covered	(T21,T25),(T25,T9),(T9,T2)

Test No:	Test#7
Test	set_price 25 insert_large_cups 1 insert_small_cups 1 coin small_cup tea coin insert_small_cups 1 coin large_cup tea insert_large_cups 1 coin sugar large_cup tea insert_large_cups 1 dispose
Transitions	T1,T4,T2,T3,T7,T21,T25,T28,T9,T7,T19,T24,T8,T7,T22,T18,T26,T8,T5
Pairs Covered	(T25,T28),(T9,T7),(T24,T8),(T8,T7),(T26,T8),(T8,T5)

Test No:	Test#8
Test	set_price 25 coin cancel insert_small_cups 1 coin cancel coin cancel set_price 50 coin coin cancel coin coin cancel dispose
Transitions	T1,T4,T7,T10,T3,T7,T10,T7,T10,T4,T6,T7,T10,T6,T7,T10,T5
Pairs Covered	(T10,T3),(T10,T7),(T10,T6),(T10,T5)

Test No:	Test#9
Test	set_price 25 insert_small_cups 5 coin small_cup tea coin small_cup tea insert_small_cups 2 coin small_cup tea insert_large_cups 5 set_price 50 coin coin small_cup tea coin coin small_cup tea set_price 25 coin small_cup tea dispose
Transitions	T1,T4,T3,T7,T21,T11,T7,T21,T11,T3,T7,T21,T11,T2,T4,T6,T7,T21,T11,T6,T7,T21,T11,T4, T7,T21,T11,T5
Pairs Covered	(T21,T11),(T11,T7),(T11,T3),(T11,T2),(T11,T4),(T11,T5),(T11,T6)

Test No:	Test#10
Test	set_price 25 insert_small_cups 5 insert_large_cups 5 coin large_cup tea coin large_cup coin tea insert_large_cups 2 coin large_cup tea insert_small_cups 5 set_price 50 coin coin large_cup tea coin coin large_cup tea set_price 25 coin large_cup tea dispose
Transitions	T1,T4,T3,T2,T7,T19,T12,T7,T19,T20,T12,T2,T7,T19,T12,T3,T4,T6,T7,T19,T12,T6,T7,T19,T12, T4,T7,T19,T12,T5
Pairs Covered	(T19,T12),(T12,T7),(T20,T12),(T12,T2),(T12,T3),(T12,T6),(T12,T4),(T12,T5)

Test No:	Test#11

Test	set_price 25 insert_large_cups 5 coin large_cup sugar tea coin large_cup sugar tea insert_large_cups 5 coin sugar large_cup tea insert_small_cups 5 coin large_cup sugar coin tea set_price 50 coin coin sugar large_cup tea coin coin sugar large_cup tea dispose
Transitions	T1,T4,T2,T7,T19,T22,T13,T7,T19,T22,T13,T2,T7,T22,T18,T13,T3,T7,T19,T22,T16,T13, T4,T6,T7,T22,T18,T13,T6,T7,T22,T18,T13,T5
Pairs Covered	(T22,T13),(T13,T2),(T13,T3),(T16,T13),(T13,T4),(T13,T6),(T13,T7)

Test No:	Test#12
Test	set_price 25 coin sugar cancel coin sugar cancel insert_large_cups 5 coin sugar coin cancel insert_small_cups 5 coin sugar small_cup cancel set_price 50 coin coin sugar large_cup cancel coin coin sugar cancel dispose
Transitions	T1,T4,T7,T22,T14,T7,T22,T14,T2,T7,T22,T16,T14,T3,T7,T22,T17,T14,T4,T6,T7,T22,T18,T14, T6,T7,T22,T14,T5
Pairs Covered	(T22,T14),(T14,T2),(T16,T14),(T14,T3),(T17,T14),(T14,T4),(T18,T14),(T14,T7),(T14,T5), (T14,T6)

Test No:	Test#13
Test	set_price 25 insert_small_cups 5 coin small_cup sugar tea coin small_cup sugar tea insert_small_cups 5 coin sugar small_cup coin tea insert_large_cups 5 coin sugar small_cup tea set_price 50 coin coin sugar small_cup tea coin coin sugar small_cup tea dispose
Transitions	T1,T4,T3,T7,T21,T22,T15,T7,T21,T22,T15,T3,T7,T22,T17,T16,T15,T2,T7,T22,T17,T15, T4,T6,T7,T22,T17,T15,T6,T7,T22,T17,T15,T5
Pairs Covered	(T22,T15),(T15,T7),(T15,T3),(T16,T15),(T15,T2),(T17,T15),(T15,T4),(T15,T6),(T15,T5)

Test No:	Test#14
Test	set_price 25 insert_large_cups 1 coin large_cup tea insert_large_cups 1 set_price 50 insert_small_cups 1 coin coin large_cup tea insert_large_cups 1 coin coin small_cup tea insert_small_cups 1 set_price 50 coin coin small_cup tea insert_small_cups 1 coin coin small_cup tea insert_small_cups 1 dispose

Transitions	T1,T4,T2,T7,T19,T24,T8,T4,T3,T6,T7,T19,T24,T8,T6,T7,T21,T25,T9, T4,T6,T7,T21,T25,T9,T6,T7,T21,T25,T9,T5
Pairs Covered	(T8,T4),(T3,T6),(T8,T6),(T9,T4),(T9,T6),(T9,T5)

Test No:	Test#15
Test	set_price 25 insert_small_cups 1 coin coin cancel coin small_cup coin tea insert_small_cups 5 insert_large_cups 1 coin large_cup coin tea insert_large_cups 1 coin small_cup cancel coin small_cup coin tea set_price 50 dispose
Transitions	T1,T4,T3,T7,T20,T10,T7,T21,T20,T25,T9,T2,T7,T19,T20,T24,T8,T7,T21,T10, T7,T21,T20,T11,T4,T5
Pairs Covered	(T20,T10),(T21,T10),(T4,T5),(T20,T25),(T20,T24),(T20,T11)

Test No:	Test#16
Test	set_price 25 insert_small_cups 1 insert_large_cups 1 coin sugar sugar sugar sugar cancel coin sugar large_cup sugar tea insert_large_cups 5 coin sugar small_cup sugar tea insert_small_cups 5 coin sugar large_cup sugar tea coin sugar small_cup sugar tea set_price 50 coin dispose
Transitions	T1,T4,T3,T2,T7,T22,T23,T22,T23,T10,T7,T22,T18,T23,T24,T8,T7,T22,T17,T23,T25,T9, T7,T22,T18,T23,T12,T7,T22,T17,T23,T11,T4,T6,T5
Pairs Covered	(T6,T5),(T23,T10),(T23,T11),(T23,T12),(T23,T22),(T23,T24),(T23,T25)

Test No:	Test#17
Test	set_price 25 insert_large_cups 1 insert_small_cups 1 coin sugar small_cup coin tea insert_small_cups 5 coin sugar large_cup coin tea insert_large_cups 1 coin sugar small_cup sugar cancel coin sugar large_cup sugar cancel coin large_cup sugar tea insert_large_cups 1 dispose
Transitions	T1,T4,T2,T3,T7,T22,T17,T16,T27,T9,T7,T22,T18,T16,T26,T8,T7,T22,T17,T23,T10, T7,T22,T18,T23,T10,T7,T19,T22,T26,T8,T5
Pairs Covered	(T17,T23),(T16,T26),(T16,T27),(T18,T23),(T22,T26),(T27,T9)

Test No:	Test#18
Test	insert_large_cups 5 set_price 25 insert_small_cups 1 coin small_cup sugar tea insert_small_cups 1 dispose
Transitions	T1,T2,T4,T3,T7,T21,T22,T27,T9,T5
Pairs Covered	(T1,T2),(T22,T27)

Test No:	Test#19
Test	insert_small_cups 5 dispose
Transitions	T1,T3,T5
Pairs Covered	(T1,T3)

Test No:	Test#20
Test	dispose
Transitions	T1,T5
Pairs Covered	(T1,T5)

Test No:	Test#21
Test	coin dispose
Transitions	T1,T5
Pairs Covered	None

2.3. Covered Transition Pairs:

The Pairs underlined are Executable and the Pairs marked in Bold are non-executable and the explanation for the same is given in 2.5.

State	Incoming	Outgoing	Transition Pairs
Idle	T1,T2,T3,T4,T6, T8,T9,T10,T11, T12,T13,T14,T15	T2,T3,T4, T5,T6,T7	(T1,T2),(T1,T3),(T1,T4),(T1,T5),(**T1,T6**),(**T1,T7**), (T2,T2),(T2,T3),(T2,T4),(T2,T5),(T2,T6),(T2,T7), (T3,T2),(T3,T3),(T3,T4),(T3,T5),(T3,T6),(T3,T7), (T4,T2),(T4,T3),(T4,T4),(T4,T5),(T4,T6),(T4,T7), (T6,T2),(T6,T3),(T6,T4),(T6,T5),(T6,T6),(T6,T7), (T8,T2),(T8,T3),(T8,T4),(T8,T5),(T8,T6),(T8,T7), (T9,T2),(T9,T3),(T9,T4),(T9,T5),(T9,T6),(T9,T7), (T10,T2),(T10,T3),(T10,T4),(T10,T5),(T10,T6),(T10,T7), (T11,T2),(T11,T3),(T11,T4),(T11,T5),(T11,T6),(T11,T7), (T12,T2),(T12,T3),(T12,T4),(T12,T5),(T12,T6),(T12,T7), (T13,T2),(T13,T3),(T13,T4),(T13,T5),(T13,T6),(T13,T7), (T14,T2),(T14,T3),(T14,T4),(T14,T5),(T14,T6),(T14,T7), (T15,T2),(T15,T3),(T15,T4),(T15,T5),(T15,T6),(T15,T7),
Coins Inserted	T7,T19,T20, T21,T23	T10,T11,T12, T19,T20,T21, T22,T24,T25	(T7,T10),(**T7,T11**),(**T7,T12**),(T7,T19),(T7,T20), (T7,T21),(T7,T22),(**T7,T24**),(**T7,T25**) (T19,T10),(**T19,T11**),(T19,T12),(T19,T19),(T19,T20), (T19,T21),(T19,T22),(T19,T24),(**T19,T25**) (T20,T10),(T20,T11),(T20,T12),(T20,T19),(T20,T20), (T20,T21),(T20,T22),(T20,T24),(T20,T25) (T21,T10),(T21,T11),(**T21,T12**),(T21,T19),(T21,T20), (T21,T21),(T21,T22),(**T21,T24**),(T21,T25) (T23,T10),(T23,T11),(T23,T12),(T23,T19),(T23,T20), (T23,T21),(T23,T22),(T23,T24),(T23,T25)
Sugar	T16,T17, T18,T22	T13,T14,T15, T16,T17,T18, T23,T26,T27	(T16,T13),(T16,T14),(T16,T15),(T16,T16),(T16,T17), (T16,T18),(T16,T23),(T16,T26),(T16,T27) (**T17,T13**),(T17,T14),(T17,T15),(T17,T16),(T17,T17), (T17,T18),(T17,T23),(**T17,T26**),(T17,T27) (T18,T13),(T18,T14),(**T18,T15**),(T18,T16),(T18,T17), (T18,T18),(T18,T23),(T18,T26),(**T18,T27**) (T22,T13),(T22,T14),(T22,T15),(T22,T16),(T22,T17), (T22,T18),(T22,T23),(T22,T26),(T22,T27)
No Small Cups	T25,T27,T28	T9,T28	(T25,T9),(T25,T28),(T27,T9),(T27,T28), (T28,T9),(T28,T28)

No Large Cups	T24,T26,T29	T8,T29	(T24,T8),(T24,T29), (T26,T8),(T26,T29), (T29,T8),(T29,T29),

2.4. Corresponding Test Case For Every Transition Pair:

State 'idle'											
Pair	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*
(T1,T2)	18	(T1,T3)	19	(T1,T4)	1	(T1,T5)	20	(T1,T6)	NE	(T1,T7)	NE
(T2,T2)	1	(T2,T3)	1	(T2,T4)	1	(T2,T5)	2	(T2,T6)	1	(T2,T7)	3
(T3,T2)	1	(T3,T3)	1	(T3,T4)	1	(T3,T5)	4	(T3,T6)	14	(T3,T7)	1
(T4,T2)	1	(T4,T3)	1	(T4,T4)	1	(T4,T5)	15	(T4,T6)	1	(T4,T7)	2
(T6,T2)	3	(T6,T3)	1	(T6,T4)	1	(T6,T5)	16	(T6,T6)	1	(T6,T7)	2
(T8,T2)	3	(T8,T3)	4	(T8,T4)	14	(T8,T5)	7	(T8,T6)	14	(T8,T7)	7
(T9,T2)	5	(T9,T3)	5	(T9,T4)	14	(T9,T5)	14	(T9,T6)	14	(T9,T7)	7
(T10,T2)	2	(T10,T3)	8	(T10,T4)	2	(T10,T5)	8	(T10,T6)	8	(T10,T7)	8
(T11,T2)	9	(T11,T3)	9	(T11,T4)	9	(T11,T5)	9	(T11,T6)	9	(T11,T7)	9
(T12,T2)	10	(T12,T3)	10	(T12,T4)	10	(T12,T5)	10	(T12,T6)	10	(T12,T7)	10
(T13,T2)	11	(T13,T3)	11	(T13,T4)	11	(T13,T5)	1	(T13,T6)	11	(T13,T7)	11
(T14,T2)	12	(T14,T3)	12	(T14,T4)	12	(T14,T5)	12	(T14,T6)	12	(T14,T7)	12
(T15,T2)	13	(T15,T3)	13	(T15,T4)	13	(T15,T5)	13	(T15,T6)	13	(T15,T7)	13

State 'coins inserted'									
Pair	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*
(T7,T10)	2	(T19,T10)	2	(T20,T10)	15	(T21,T10)	15	(T23,T10)	16
(T7,T11)	NE	(T19,T11)	NE	(T20,T11)	15	(T21,T11)	9	(T23,T11)	16
(T7,T12)	NE	(T19,T12)	10	(T20,T12)	10	(T21,T12)	NE	(T23,T12)	16
(T7,T19)	2	(T19,T19)	2	(T20,T19)	2	(T21,T19)	4	(T23,T19)	2
(T7,T20)	1	(T19,T20)	2	(T20,T20)	1	(T21,T20)	2	(T23,T20)	2

(T7,T21)	4	(T19,T21)	2	(T20,T21)	1	(T21,T21)	1	(T23,T21)	2
(T7,T22)	5	(T19,T22)	4	(T20,T22)	2	(T21,T22)	1	(T23,T22)	16
(T7,T24)	NE	(T19,T24)	3	(T20,T24)	15	(T21,T24)	NE	(T23,T24)	16
(T7,T25)	NE	(T19,T25)	NE	(T20,T25)	15	(T21,T25)	6	(T23,T25)	16

State 'sugar'							
Pair	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*	*Pair*	*Test#*
(T16,T13)	11	(T17,T13)	NE	(T18,T13)	1	(T22,T13)	11
(T16,T14)	12	(T17,T14)	12	(T18,T14)	12	(T22,T14)	12
(T16,T15)	13	(T17,T15)	13	(T18,T15)	NE	(T22,T15)	13
(T16,T16)	2	(T17,T16)	2	(T18,T16)	4	(T22,T16)	2
(T16,T17)	2	(T17,T17)	1	(T18,T17)	2	(T22,T17)	1
(T16,T18)	4	(T17,T18)	1	(T18,T18)	2	(T22,T18)	4
(T16,T23)	2	(T17,T23)	17	(T18,T23)	17	(T22,T23)	2
(T16,T26)	17	(T17,T26)	NE	(T18,T26)	4	(T22,T26)	17
(T16,T27)	17	(T17,T27)	5	(T18,T27)	NE	(T22,T27)	18

State 'no small cups'		State 'no large cups'	
Pair	*Test#*	*Pair*	*Test#*
(T25,T9)	6	(T24,T8)	7
(T25,T28)	7	(T24,T29)	3
(T27,T9)	17	(T26,T8)	7
(T27,T28)	5	(T26,T29)	4
(T28,T9)	5	(T29,T8)	3
(T28,T28)	5	(T29,T29)	3

2.5. Explanation for Non-Executable Transition Pairs:

1. (T7, T11): At the transition T7, the value of s and t is set to 0. When T11 occurs, the value of s should be 2 (small_cup) at that time, which is never set if we go from T7 to T11 directly. Thus, this pair is non-executable.

2. (T7, T12): At the transition T7, the value of s and t is set to 0. When T12 occurs, the value of s should be 1 (large_cup) at that time, which is never set if we go from T7 to T12 directly. Thus, this pair is non-executable.

3. (T7, T24): At the transition T7, the value of s and t is set to 0. When T24 occurs, the value of s should be 1 (large_cup) at that time, which is never set if we go from T7 to T24 directly. Thus, this pair is non-executable.

4. (T7, T25): At the transition T7, the value of s and t is set to 0. When T25 occurs, the value of s should be 2 (small_cup) at that time, which is never set if we go from T7 to T25 directly. Thus, this pair is non-executable.

5. (T19, T11): At Transition T19, large_cup is selected and value of s is set to 1. Transition T11 requires the value of s to be 2 (small_cup). This never happens if we go directly from T19 to T11. Thus, this pair is non-executable.

6. (T19, T25): At Transition T19, large_cup is selected and value of s is set to 1. Transition T25 requires the value of s to be 2 (small_cup). This never happens if we go directly from T19 to T25. Thus, this pair is non-executable.

7. (T21, T12): At Transition T21, small_cup is selected and value of s is set to 2. Transition T12 requires the value of s to be 1 (large_cup). This never happens if we go directly from T21 to T12. Thus, this pair is non-executable.

8. (T21, T24): At Transition T21, small_cup is selected and value of s is set to 2. Transition T24 requires the value of s to be 1 (large_cup). This never happens if we go directly from T21 to T24. Thus, this pair is non-executable.

9. (T17, T13): At Transition T17, small_cup is selected and value of s is set to 2. Transition T13 requires the value of s to be 1 (large_cup). This never happens if we go directly from T17 to T13. Thus, this pair is non-executable.

10. (T17, T26): At Transition T17, small_cup is selected and value of s is set to 2. Transition T26 requires the value of s to be 1 (large_cup). This never happens if we go directly from T17 to T26. Thus, this pair is non-executable.

11. (T18, T15): At Transition T18, large_cup is selected and value of s is set to 1. Transition T15 requires the value of s to be 2 (small_cup). This never happens if we go directly from T18 to T15. Thus, this pair is non-executable.

12. (T18, T27): At Transition T18, large_cup is selected and value of s is set to 1. Transition T27 requires the value of s to be 2 (small_cup). This never happens if we go directly from T18 to T27. Thus, this pair is non-executable.

13. (T1, T7): When T1 occurs, the constructor is called and all the variables are set to 0. In order for T7 to occur, the condition price>0 should be true. Which in this pair is false. So this pair is non-executable.

14. (T1, T6): When T1 occurs, the constructor is called and all the variables are set to 0. Now price=0 and for transition T6 to occur, price should be greater than or equal to 25 which in this case is false. So this pair is non-executable.

3. Ghost/Default Transition Testing:

The additional test cases for ghost transition are as follows:

Test No:	Test#22
Test	small_cup large_cup sugar tea insert_large_cups 0 insert_small_cups 0 set_price 0 cancel coin dispose

Test No:	Test#23
Test	set_price 25 insert_large_cups 5 coin tea insert_large_cups 5 insert_small_cups 5 set_price 25 dispose cancel dispose

Test No:	Test#24
Test	set_price 25 insert_small_cups 5 coin sugar tea insert_large_cups 0 insert_small_cups 0 set_price 25 dispose cancel dispose

Test No:	Test#25
Test	set_price 25 insert_small_cups 1 coin small_cup tea small_cup large_cup sugar tea insert_large_cups 5 insert_small_cups 0 set_price 0 cancel dispose insert_small_cups 5 dispose

Test No:	Test#26
Test	set_price 25 insert_large_cups 1 coin large_cup tea small_cup large_cup sugar tea insert_large_cups 0 insert_small_cups 0 set_price 25 cancel dispose insert_large_cups 5 dispose

I have divided the ghost/default transitions state-wise. They are as follows:

1. State 'idle':

Total Ghost Transitions: 8	
Ghost Transitions	*Test#*
small_cup() large_cup() sugar() tea() insert_large_cups(int n) [n<=0] insert_small_cups(int n) [n<=0]	Test#22

set_price(int p) [p<=0] cancel() coin() [price<=0]	

2. State 'coins inserted':

Total Ghost Transitions: 5	
Ghost Transitions	*Test#*
tea() [k<1, s!=1, s!=2, k1<1] insert_large_cups(int n) insert_small_cups(int n) set_price(int p) dispose()	Test#23

3. State 'sugar':

Total Ghost Transitions: 5	
Ghost Transitions	*Test#*
tea() [k<1, s!=1, s!=2, k1<1] insert_large_cups(int n) insert_small_cups(int n) set_price(int p) dispose()	Test#24

4. State 'no small cups':

Total Ghost Transitions: 9	
Ghost Transitions	*Test#*
small_cup large_cup sugar() tea() insert_large_cups(int n) insert_small_cups(int n) [n<=0] set_price(int p) cancel() dispose()	Test#25

5. State 'no large cups':

Total Ghost Transitions: 9	
Ghost Transitions	*Test#*

small_cup large_cup sugar() tea() insert_large_cups(int n) [n<=0] insert_small_cups(int n) set_price(int p) cancel() dispose()	Test#26

4. Multiple Condition Testing:

The additional test cases for Multiple Condition Testing are as follows:

Test No:	Test#27
Test	set_price 25 coin tea sugar tea sugar cancel dispose

Test No:	Test#28
Test	set_price 25 insert_large_cups 1 insert_small_cups 1 coin tea sugar tea cancel dispose

Test No:	Test#29
Test	set_price 25 coin large_cup tea sugar tea cancel dispose

Test No:	Test#30
Test	set_price 25 coin small_cup tea sugar tea cancel dispose

Test No:	Test#31
Test	set_price 25 insert_large_cups 5 coin tea cancel insert_small_cups 5 coin tea cancel dispose

Test No:	Test#32
Test	set_price 25 coin large_cup tea small_cup tea cancel dispose

Test No:	Test#33
Test	set_price 25 insert_large_cups 1 insert_small_cups 1 coin tea sugar tea cancel dispose

Test No:	Test#34
Test	set_price 25 dispose coin dispose

Identified multiple conditions and their corresponding test cases are as follows. There are some multiple conditions which are non-executable. The explanation of all such non-executable multiple conditions are also covered below.

1. coin():

1.1. Condition: (x == 1)	
(x=1)	Test #
T	Test#1
F	Test#2

1.2. Condition: (t + 25 >= price) && (price > 0)		
(t + 25 >= price)	(price > 0)	Test #
T	T	Test#1
T	F	Test#22
F	T	Test#1
F	F	Non-Executable: If (price>0) is false, (t+25>=price) will never be false because (t+25) will result in 25 which will be greater than price if it is less than 0

1.3. Condition: (t + 25 < price)	
(t + 25 < price)	Test #
T	Test#1
F	Test#22

1.4. Condition: ((x > 1) && (x < 6))		
(x > 1)	(x < 6)	Test #
T	T	Test#1
T	F	Test#34
F	T	Non-Executable: The condition x<1 and x<6 leaves only one value for x and that is x=1. But when x=1, it will always go into the condition (x==1) and never execute this condition.
F	F	Non-Executable: There is no value of x possible such that x<=1 and x>=6

2. small_cup():

2.1. Condition: ((x == 2) || (x == 3))		
(x == 2)	(x == 3)	Test #
T	T	Non-Executable: There is only 1 variable x for state and it can either be x==2 (coins inserted) or x==3 (sugar). Both can never be true at the same time (Can never have 2 states at the same time).
T	F	Test#1
F	T	Test#1
F	F	Test#22

3. large_cup():

3.2. Condition: ((x == 2) || (x == 3))		
(x == 2)	(x == 3)	Test #
T	T	Non-Executable: There is only 1 variable x for state and it can either be x==2 (coins inserted) or x==3 (sugar). Both can never be true at the same time (Can never have 2 states at the same time).
T	F	Test#2
F	T	Test#1
F	F	Test#22

4. sugar():

4.1. Condition: ((x == 2) || (x == 3))		
(x == 2)	(x == 3)	Test #
T	T	Non-Executable: There is only 1 variable x for state and it can either be x==2 (coins inserted) or x==3 (sugar). Both can never be true at the same time (Can never have 2 states at the same time).
T	F	Test#1
F	T	Test#2
F	F	Test#22

4.2. Condition: (x == 2)	
(x == 2)	Test #

T	Test#1
F	Test#2

5. tea():

5.1. Condition: ((x == 2) || (x == 3))		
(x == 2)	(x == 3)	Test #
T	T	Non-Executable: There is only 1 variable x for state and it can either be x==2 (coins inserted) or x==3 (sugar). Both can never be true at the same time (Can never have 2 states at the same time).
T	F	Test#3
F	T	Test#1
F	F	Test#22

5.2. Condition: ((x == 2) && (k1 > 1) && (s == 2))			
(x == 2)	(k1 > 1)	(s == 2)	Test #
T	T	T	Test#9
T	T	F	Test#31
T	F	T	Test#32
T	F	F	Test#27
F	T	T	Test#13
F	T	F	Test#33
F	F	T	Test#30
F	F	F	Test#27

5.3. Condition: ((x == 2) && (k > 1) && (s == 1))			
(x == 2)	(k > 1)	(s == 1)	Test #
T	T	T	Test#10
T	T	F	Test#31
T	F	T	Test#32
T	F	F	Test#27
F	T	T	Test#1
F	T	F	Test#31
F	F	T	Test#32
F	F	F	Test#27

5.4. Condition: ((x == 2) && (k == 1) && (s == 1))

(x == 2)	(k == 1)	(s == 1)	Test #
T	T	T	Test#3
T	T	F	Test#28
T	F	T	Test#29
T	F	F	Test#27
F	T	T	Test#4
F	T	F	Test#33
F	F	T	Test#29
F	F	F	Test#27

5.5. Condition: ((x == 2) && (k1 == 1) && (s == 2))			
(x == 2)	(k1 == 1)	(s == 2)	Test #
T	T	T	Test#6
T	T	F	Test#28
T	F	T	Test#30
T	F	F	Test#27
F	T	T	Test#5
F	T	F	Test#33
F	F	T	Test#30
F	F	F	Test#27

5.6. Condition: ((x == 3) && (k1 == 1) && (s == 2))			
(x == 3)	(k1 == 1)	(s == 2)	Test #
T	T	T	Test#5
T	T	F	Test#28
T	F	T	Test#30
T	F	F	Test#27
F	T	T	Non-Executable: This condition will result into (x==2) && (k1==1) && (s==2), which will get checked first when code enters the first 'if' of tea() method and it will definitely go in the 'if' loop. So, this condition will never be evaluated and so it is non-executable
F	T	F	Test#33
F	F	T	Test#32
F	F	F	Test#27

5.7. Condition: ((x == 3) && (k == 1) && (s == 1))			
(x == 3)	(k == 1)	(s == 1)	Test #
T	T	T	Test#4

T	T	F	Test#28
T	F	T	Test#29
T	F	F	Test#27
F	T	T	Non-Executable: This condition will result into (x==2) && (k==1) && (s==1), which will get checked first when code enters the first 'if' of tea() method and it will definitely go in the 'if' loop. So, this condition will never be evaluated and so it is non-executable
F	T	F	Test#33
F	F	T	Test#32
F	F	F	Test#27

5.8. Condition: ((x == 3) && (k1 > 1) && (s == 2))			
(x == 3)	(k1 > 1)	(s == 2)	Test #
T	T	T	Test#13
T	T	F	Test#28
T	F	T	Test#30
T	F	F	Test#27
F	T	T	Non-Executable: This condition will result into (x==2) && (k1>1) && (s==2), which will get checked first when code enters the first 'if' of tea() method and it will definitely go in the 'if' loop. So, this condition will never be evaluated and so it is non-executable
F	T	F	Test#31
F	F	T	Test#32
F	F	F	Test#27

5.9. Condition: ((x == 3) && (k > 1) && (s == 1))			
(x == 3)	(k > 1)	(s == 1)	Test #
T	T	T	Test#1
T	T	F	Test#28
T	F	T	Test#30
T	F	F	Test#27
F	T	T	Non-Executable: This condition will result into (x==2) && (k>1) && (s==1), which will get checked first when code enters the first 'if' of tea() method and it will definitely go in the 'if' loop. So, this

			condition will never be evaluated and so it is non-executable
F	T	F	Test#31
F	F	T	Test#32
F	F	F	Test#27

6. insert_large_cups(int n):

6.1. Condition: ((x == 1) && (n > 0))		
(x == 1)	(n > 0)	Test #
T	T	Test#1
T	F	Test#22
F	T	Test#3
F	F	Test#26

6.2. Condition: ((x == 5) && (n > 0))		
(x == 5)	(n > 0)	Test #
T	T	Test#3
T	F	Test#26
F	T	Test#23
F	F	Test#22

7. insert_small_cups(int n):

7.1. Condition: ((x == 1) && (n > 0))		
(x == 1)	(n > 0)	Test #
T	T	Test#1
T	F	Test#22
F	T	Test#5
F	F	Test#24

7.2. Condition: ((x == 4) && (n > 0))		
(x == 4)	(n > 0)	Test #
T	T	Test#5
T	F	Test#25
F	T	Test#23
F	F	Test#26

8. set_price(int p):

8.1. Condition: ((x == 1) && (p > 0))

(x == 1)	(p > 0)	Test #
T	T	Test#1
T	F	Test#22
F	T	Test#23
F	F	Test#25

9. cancel():

9.1. Condition: ((x == 2) || (x == 3))		
(x == 2)	(x == 3)	Test #
T	T	Non-Executable: There is only 1 variable x for state and it can either be x==2 (coins inserted) or x==3 (sugar). Both can never be true at the same time (Can never have 2 states at the same time).
T	F	Test#2
F	T	Test#12
F	F	Test#22

10. dispose():

10.1. Condition: ((x == 1))	
(x == 1)	Test #
T	Test#1
F	Test#23

5. Test Suite:

Test#1: set_price 25 insert_large_cups 5 insert_large_cups 5 insert_small_cups 5 insert_small_cups 5 insert_large_cups 5 set_price 50 insert_small_cups 5 set_price 50 set_price 75 coin coin set_price 100 insert_large_cups 5 coin insert_small_cups 5 coin coin coin small_cup small_cup sugar small_cup small_cup large_cup tea dispose

Test#2: set_price 25 coin cancel set_price 50 coin coin large_cup large_cup coin large_cup small_cup coin small_cup sugar coin coin small_cup small_cup large_cup large_cup small_cup coin sugar coin sugar sugar small_cup sugar sugar large_cup cancel insert_large_cups 1 dispose

Test#3: set_price 50 coin insert_large_cups 1 coin large_cup tea coin coin insert_large_cups 1 insert_large_cups 1 dispose

Test#4: set_price 25 insert_large_cups 1 coin small_cup large_cup sugar large_cup coin large_cup tea coin insert_large_cups 1 insert_small_cups 1 dispose

Test#5: set_price 25 insert_large_cups 5 insert_small_cups 1 coin sugar small_cup tea coin coin insert_small_cups 1 insert_small_cups 1 dispose

Test#6: set_price 25 insert_small_cups 1 coin small_cup tea insert_small_cups 1 insert_large_cups 1 coin small_cup sugar tea insert_small_cups 1 insert_small_cups 1 dispose

Test#7: set_price 25 insert_large_cups 1 insert_small_cups 1 coin small_cup tea coin insert_small_cups 1 coin large_cup tea insert_large_cups 1 coin sugar large_cup tea insert_large_cups 1 dispose

Test#8: set_price 25 coin cancel insert_small_cups 1 coin cancel coin cancel set_price 50 coin coin cancel coin coin cancel dispose

Test#9: set_price 25 insert_small_cups 5 coin small_cup tea coin small_cup tea insert_small_cups 2 coin small_cup tea insert_large_cups 5 set_price 50 coin coin small_cup tea coin coin small_cup tea set_price 25 coin small_cup tea dispose

Test#10: set_price 25 insert_small_cups 5 insert_large_cups 5 coin large_cup tea coin large_cup coin tea insert_large_cups 2 coin large_cup tea insert_small_cups 5 set_price 50 coin coin large_cup tea coin coin large_cup tea set_price 25 coin large_cup tea dispose

Test#11: set_price 25 insert_large_cups 5 coin large_cup sugar tea coin large_cup sugar tea insert_large_cups 5 coin sugar large_cup tea insert_small_cups 5 coin large_cup sugar coin tea set_price 50 coin coin sugar large_cup tea coin coin sugar large_cup tea dispose

Test#12: set_price 25 coin sugar cancel coin sugar cancel insert_large_cups 5 coin sugar coin cancel insert_small_cups 5 coin sugar small_cup cancel set_price 50 coin coin sugar large_cup cancel coin coin sugar cancel dispose

Test#13: set_price 25 insert_small_cups 5 coin small_cup sugar tea coin small_cup sugar tea insert_small_cups 5 coin sugar small_cup coin tea insert_large_cups 5 coin sugar small_cup tea set_price 50 coin coin sugar small_cup tea coin coin sugar small_cup tea dispose

Test#14: set_price 25 insert_large_cups 1 coin large_cup tea insert_large_cups 1 set_price 50 insert_small_cups 1 coin coin large_cup tea insert_large_cups 1 coin coin small_cup tea insert_small_cups 1 set_price 50 coin coin small_cup tea insert_small_cups 1 coin coin small_cup tea insert_small_cups 1 dispose

Test#15: set_price 25 insert_small_cups 1 coin coin cancel coin small_cup coin tea insert_small_cups 5 insert_large_cups 1 coin large_cup coin tea insert_large_cups 1 coin small_cup cancel coin small_cup coin tea set_price 50 dispose

Test#16: set_price 25 insert_small_cups 1 insert_large_cups 1 coin sugar sugar sugar sugar cancel coin sugar large_cup sugar tea insert_large_cups 5 coin sugar small_cup sugar tea insert_small_cups 5 coin sugar large_cup sugar tea coin sugar small_cup sugar tea set_price 50 coin dispose

Test#17: set_price 25 insert_large_cups 1 insert_small_cups 1 coin sugar small_cup coin tea insert_small_cups 5 coin sugar large_cup coin tea insert_large_cups 1 coin sugar small_cup sugar cancel coin sugar large_cup sugar cancel coin large_cup sugar tea insert_large_cups 1 dispose

Test#18: insert_large_cups 5 set_price 25 insert_small_cups 1 coin small_cup sugar tea insert_small_cups 1 dispose

Test#19: insert_small_cups 5 dispose

Test#20: dispose

Test#21: coin dispose

Test#22: small_cup large_cup sugar tea insert_large_cups 0 insert_small_cups 0 set_price 0 cancel coin dispose

Test#23: set_price 25 insert_large_cups 5 coin tea insert_large_cups 5 insert_small_cups 5 set_price 25 dispose cancel dispose

Test#24: set_price 25 insert_small_cups 5 coin sugar tea insert_large_cups 0 insert_small_cups 0 set_price 25 dispose cancel dispose

Test#25: set_price 25 insert_small_cups 1 coin small_cup tea small_cup large_cup sugar tea insert_large_cups 5 insert_small_cups 0 set_price 0 cancel dispose insert_small_cups 5 dispose

Test#26: set_price 25 insert_large_cups 1 coin large_cup tea small_cup large_cup sugar tea insert_large_cups 0 insert_small_cups 0 set_price 25 cancel dispose insert_large_cups 5 dispose

Test#27: set_price 25 coin tea sugar tea sugar cancel dispose

Test#28: set_price 25 insert_large_cups 1 insert_small_cups 1 coin tea sugar tea cancel dispose

Test#29: set_price 25 coin large_cup tea sugar tea cancel dispose

Test#30: set_price 25 coin small_cup tea sugar tea cancel dispose

Test#31: set_price 25 insert_large_cups 5 coin tea cancel insert_small_cups 5 coin tea cancel dispose

Test#32: set_price 25 coin large_cup tea small_cup tea cancel dispose

Test#33: set_price 25 insert_large_cups 1 insert_small_cups 1 coin tea sugar tea cancel dispose

Test#34: set_price 25 dispose coin dispose

$$ $$

6. Results of Test Execution:

Each and every test case from the test suite was tested with the test driver and the results produced by the VendingMachine class were recorded. The values of all the variables were recorded for each transition (method invoked) for every test case. All the recorded variables, states, expected results and actual results are documented as follows:

Test#1:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 5	1	25	5	0	0	0	idle	pass	pass
insert_large_cups 5	1	25	10	0	0	0	idle	pass	pass
insert_small_cups 5	1	25	10	5	0	0	idle	pass	pass
insert_small_cups 5	1	25	10	10	0	0	idle	pass	pass
insert_large_cups 5	1	25	15	10	0	0	idle	pass	pass
set_price 50	1	50	15	10	0	0	idle	pass	pass
insert_small_cups 5	1	50	15	15	0	0	idle	pass	pass
set_price 50	1	50	15	15	0	0	idle	pass	pass
set_price 75	1	75	15	15	0	0	idle	pass	pass
coin	1	75	15	15	0	25	idle	pass	pass
coin	1	75	15	15	0	50	idle	pass	pass
set_price 100	1	100	15	15	0	50	idle	pass	pass
insert_large_cups 5	1	100	20	15	0	50	idle	pass	pass
coin	1	100	20	15	0	75	idle	pass	pass
insert_small_cups 5	1	100	20	20	0	75	idle	pass	pass
coin	2	100	20	20	0	0	coins inserted	pass	pass
coin	2	100	20	20	0	0	coins inserted	pass	pass
coin	2	100	20	20	0	0	coins inserted	pass	pass
small_cup	2	100	20	20	2	0	coins inserted	pass	pass

small_cup	2	100	20	20	2	0	coins inserted	pass	pass
sugar	3	100	20	20	2	0	sugar	pass	pass
small_cup	3	100	20	20	2	0	sugar	pass	pass
small_cup	3	100	20	20	2	0	sugar	pass	pass
large_cup	3	100	20	20	1	0	sugar	pass	pass
tea	1	100	19	20	1	0	idle	pass	pass
dispose	6	100	19	20	1	0	shut down	pass	pass

Test#2:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
coin	2	25	0	0	0	0	coins inserted	pass	pass
cancel	1	25	0	0	0	0	idle	pass	pass
set_price 50	1	50	0	0	0	0	idle	pass	pass
coin	1	50	0	0	0	25	idle	pass	pass
coin	2	50	0	0	0	0	coins inserted	pass	pass
large_cup	2	50	0	0	1	0	coins inserted	pass	pass
large_cup	2	50	0	0	1	0	coins inserted	pass	pass
coin	2	50	0	0	1	0	coins inserted	pass	pass
large_cup	2	50	0	0	1	0	coins inserted	pass	pass
small_cup	2	50	0	0	2	0	coins inserted	pass	pass
coin	2	50	0	0	2	0	coins inserted	pass	pass
small_cup	2	50	0	0	2	0	coins inserted	pass	pass
sugar	3	50	0	0	2	0	sugar	pass	pass
coin	3	50	0	0	2	0	sugar	pass	pass
coin	3	50	0	0	2	0	sugar	pass	pass
small_cup	3	50	0	0	2	0	sugar	pass	pass

small_cup	3	50	0	0	2	0	sugar	pass	pass
large_cup	3	50	0	0	1	0	sugar	pass	pass
large_cup	3	50	0	0	1	0	sugar	pass	pass
small_cup	3	50	0	0	2	0	sugar	pass	pass
coin	3	50	0	0	2	0	sugar	pass	pass
sugar	2	50	0	0	2	0	coins inserted	pass	pass
coin	2	50	0	0	2	0	coins inserted	pass	pass
sugar	3	50	0	0	2	0	sugar	pass	pass
sugar	2	50	0	0	2	0	coins inserted	pass	pass
small_cup	2	50	0	0	2	0	coins inserted	pass	pass
sugar	3	50	0	0	2	0	sugar	pass	pass
sugar	2	50	0	0	2	0	coins inserted	pass	pass
large_cup	2	50	0	0	1	0	coins inserted	pass	pass
cancel	1	50	0	0	1	0	idle	pass	pass
insert_large_cups 1	1	50	1	0	1	0	idle	pass	pass
dispose	6	50	1	0	1	0	shut down	pass	pass

Test#3:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 50	1	50	0	0	0	0	idle	pass	pass
coin	1	50	0	0	0	25	idle	pass	pass
insert_large_cups 1	1	50	1	0	0	25	idle	pass	pass
coin	2	50	1	0	0	0	coins inserted	pass	pass
large_cup	2	50	1	0	1	0	coins inserted	pass	pass
tea	5	50	0	0	1	0	no large cups	pass	pass
coin	5	50	0	0	1	0	no large cups	pass	pass
coin	5	50	0	0	1	0	no large cups	pass	pass

insert_large_cups 1	1	50	1	0	1	0	idle	pass	pass
insert_large_cups 1	1	50	2	0	1	0	idle	pass	pass
dispose	6	50	2	0	1	0	shut down	pass	pass

Test#4:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 1	1	25	1	0	0	0	idle	pass	pass
coin	2	25	1	0	0	0	coins inserted	pass	pass
small_cup	2	25	1	0	2	0	coins inserted	pass	pass
large_cup	2	25	1	0	1	0	coins inserted	pass	pass
sugar	3	25	1	0	1	0	sugar	pass	pass
large_cup	3	25	1	0	1	0	sugar	pass	pass
coin	3	25	1	0	1	0	sugar	pass	pass
large_cup	3	25	1	0	1	0	sugar	pass	pass
tea	5	25	0	0	1	0	no large cups	pass	pass
coin	5	25	0	0	1	0	no large cups	pass	pass
insert_large_cups 1	1	25	1	0	1	0	idle	pass	pass
insert_small_cups 1	1	25	1	1	1	0	idle	pass	pass
dispose	6	25	1	1	1	0	shut down	pass	pass

Test#5:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass

insert_large_cups 5	1	25	5	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	5	1	0	0	idle	pass	pass
coin	2	25	5	1	0	0	coins inserted	pass	pass
sugar	3	25	5	1	0	0	sugar	pass	pass
small_cup	3	25	5	1	2	0	sugar	pass	pass
tea	4	25	5	0	2	0	no small cups	pass	pass
coin	4	25	5	0	2	0	no small cups	pass	pass
coin	4	25	5	0	2	0	no small cups	pass	pass
insert_small_cups 1	1	25	5	1	2	0	idle	pass	pass
insert_small_cups 1	1	25	5	2	2	0	idle	pass	pass
dispose	6	25	5	2	2	0	shut down	pass	pass

Test#6:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	0	1	0	0	idle	pass	pass
coin	2	25	0	1	0	0	coins inserted	pass	pass
small_cup	2	25	0	1	2	0	coins inserted	pass	pass
tea	4	25	0	0	2	0	no small cups	pass	pass
insert_small_cups 1	1	25	0	1	2	0	idle	pass	pass
insert_large_cups 1	1	25	1	1	2	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
small_cup	2	25	1	1	2	0	coins inserted	pass	pass
sugar	3	25	1	1	2	0	sugar	pass	pass
tea	4	25	1	0	2	0	no small cups	pass	pass
insert_small_cups 1	1	25	1	1	2	0	idle	pass	pass
insert_small_cups 1	1	25	1	2	2	0	idle	pass	pass

dispose	6	25	1	2	2	0	shut down	pass	pass

Test#7:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 1	1	25	1	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	1	1	0	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
small_cup	2	25	1	1	2	0	coin inserted	pass	pass
tea	4	25	1	0	2	0	no small cups	pass	pass
coin	4	25	1	0	2	0	no small cups	pass	pass
insert_small_cups 1	1	25	1	1	2	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
large_cup	2	25	1	1	1	0	coins inserted	pass	pass
tea	5	25	0	1	1	0	no large cups	pass	pass
insert_large_cups 1	1	25	1	1	1	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
sugar	3	25	1	1	0	0	sugar	pass	pass
large_cup	3	25	1	1	1	0	sugar	pass	pass
tea	5	25	0	1	1	0	no large cups	pass	pass
insert_large_cups 1	1	25	1	1	1	0	idle	pass	pass
dispose	6	25	1	1	1	0	shut down	pass	pass

Test#8:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	

vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
coin	2	25	0	0	0	0	coins inserted	pass	pass
cancel	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	0	1	0	0	idle	pass	pass
coin	2	25	0	1	0	0	coins inserted	pass	pass
cancel	1	25	0	1	0	0	idle	pass	pass
coin	2	25	0	1	0	0	coins inserted	pass	pass
cancel	1	25	0	1	0	0	idle	pass	pass
set_price 50	1	50	0	1	0	0	idle	pass	pass
coin	1	50	0	1	0	25	idle	pass	pass
coin	2	50	0	1	0	0	coins inserted	pass	pass
cancel	1	50	0	1	0	0	idle	pass	pass
coin	1	50	0	1	0	25	idle	pass	pass
coin	2	50	0	1	0	0	coins inserted	pass	pass
cancel	1	50	0	1	0	0	idle	pass	pass
dispose	6	50	0	1	0	0	shut down	pass	pass

Test#9:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 5	1	25	0	5	0	0	idle	pass	pass
coin	2	25	0	5	0	0	coins inserted	pass	pass
small_cup	2	25	0	5	2	0	coins inserted	pass	pass
tea	1	25	0	4	2	0	idle	pass	pass
coin	2	25	0	4	0	0	coins inserted	pass	pass
small_cup	2	25	0	4	2	0	coins inserted	pass	pass

tea	1	25	0	3	2	0	idle	pass	pass
insert_small_cups 2	1	25	0	5	2	0	idle	pass	pass
coin	2	25	0	5	0	0	coins inserted	pass	pass
small_cup	2	25	0	5	2	0	coins inserted	pass	pass
tea	1	25	0	4	2	0	idle	pass	pass
insert_large_cups 5	1	25	5	4	2	0	idle	pass	pass
set_price 50	1	50	5	4	2	0	idle	pass	pass
coin	1	50	5	4	2	25	idle	pass	pass
coin	2	50	5	4	0	0	coins inserted	pass	pass
small_cup	2	50	5	4	2	0	coins inserted	pass	pass
tea	1	50	5	3	2	0	idle	pass	pass
coin	1	50	5	3	2	25	idle	pass	pass
coin	2	50	5	3	0	0	coins inserted	pass	pass
small_cup	2	50	5	3	2	0	coins inserted	pass	pass
tea	1	50	5	2	2	0	idle	pass	pass
set_price 25	1	25	5	2	2	0	idle	pass	pass
coin	2	25	5	2	0	0	coins inserted	pass	pass
small_cup	2	25	5	2	2	0	coins inserted	pass	pass
tea	1	25	5	1	2	0	idle	pass	pass
dispose	6	25	5	1	2	0	shut down	pass	pass

Test#10:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 5	1	25	0	5	0	0	idle	pass	pass
insert_large_cups 5	1	25	5	5	0	0	idle	pass	pass
coin	2	25	5	5	0	0	coins inserted	pass	pass

large_cup	2	25	5	5	1	0	coins inserted	pass	pass
tea	1	25	4	5	1	0	idle	pass	pass
coin	2	25	4	5	0	0	coins inserted	pass	pass
large_cup	2	25	4	5	1	0	coins inserted	pass	pass
coin	2	25	4	5	1	0	coins inserted	pass	pass
tea	1	25	3	5	1	0	idle	pass	pass
insert_large_cups 2	1	25	5	5	1	0	idle	pass	pass
coin	2	25	5	5	0	0	coins inserted	pass	pass
large_cup	2	25	5	5	1	0	coins inserted	pass	pass
tea	1	25	4	5	1	0	idle	pass	pass
insert_small_cups 5	1	25	4	10	1	0	idle	pass	pass
set_price 50	1	50	4	10	1	0	idle	pass	pass
coin	1	50	4	10	1	25	idle	pass	pass
coin	2	50	4	10	0	0	coins inserted	pass	pass
large_cup	2	50	4	10	1	0	coins inserted	pass	pass
tea	1	50	3	10	1	0	idle	pass	pass
coin	1	50	3	10	1	25	idle	pass	pass
coin	2	50	3	10	0	0	coins inserted	pass	pass
large_cup	2	50	3	10	1	0	coins inserted	pass	pass
tea	1	50	2	10	1	0	idle	pass	pass
set_price 25	1	25	2	10	1	0	idle	pass	pass
coin	2	25	2	10	0	0	coins inserted	pass	pass
large_cup	2	25	2	10	1	0	coins inserted	pass	pass
tea	1	25	1	10	1	0	idle	pass	pass
dispose	6	25	1	10	1	0	shut down	pass	pass

Test#11:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	

vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 5	1	25	5	0	0	0	idle	pass	pass
coin	2	25	5	0	0	0	coins inserted	pass	pass
large_cup	2	25	5	0	1	0	coins inserted	pass	pass
sugar	3	25	5	0	1	0	sugar	pass	pass
tea	1	25	4	0	1	0	idle	pass	pass
coin	2	25	4	0	0	0	coins inserted	pass	pass
large_cup	2	25	4	0	1	0	coins inserted	pass	pass
sugar	3	25	4	0	1	0	sugar	pass	pass
tea	1	25	3	0	1	0	idle	pass	pass
insert_large_cups 5	1	25	8	0	1	0	idle	pass	pass
coin	2	25	8	0	0	0	coins inserted	pass	pass
sugar	3	25	8	0	0	0	sugar	pass	pass
large_cup	3	25	8	0	1	0	sugar	pass	pass
tea	1	25	7	0	1	0	idle	pass	pass
insert_small_cups 5	1	25	7	5	1	0	idle	pass	pass
coin	2	25	7	5	0	0	coins inserted	pass	pass
large_cup	2	25	7	5	1	0	coins inserted	pass	pass
sugar	3	25	7	5	1	0	sugar	pass	pass
coin	3	25	7	5	1	0	sugar	pass	pass
tea	1	25	6	5	1	0	idle	pass	pass
set_price 50	1	50	6	5	1	0	idle	pass	pass
coin	1	50	6	5	1	25	idle	pass	pass
coin	2	50	6	5	0	0	coins inserted	pass	pass
sugar	3	50	6	5	0	0	sugar	pass	pass
large_cup	3	50	6	5	1	0	sugar	pass	pass
tea	1	50	5	5	1	0	idle	pass	pass
coin	1	50	5	5	1	25	idle	pass	pass

coin	2	50	5	5	0	0	coins inserted	pass	pass
sugar	3	50	5	5	0	0	sugar	pass	pass
large_cup	3	50	5	5	1	0	sugar	pass	pass
tea	1	50	4	5	1	0	idle	pass	pass
dispose	6	50	4	5	1	0	shut down	pass	pass

Test#12:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
coin	2	25	0	0	0	0	coins inserted	pass	pass
sugar	3	25	0	0	0	0	sugar	pass	pass
cancel	1	25	0	0	0	0	idle	pass	pass
coin	2	25	0	0	0	0	coins inserted	pass	pass
sugar	3	25	0	0	0	0	sugar	pass	pass
cancel	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 5	1	25	5	0	0	0	idle	pass	pass
coin	2	25	5	0	0	0	coins inserted	pass	pass
sugar	3	25	5	0	0	0	sugar	pass	pass
coin	3	25	5	0	0	0	sugar	pass	pass
cancel	1	25	5	0	0	0	idle	pass	pass
insert_small_cups 5	1	25	5	5	0	0	idle	pass	pass
coin	2	25	5	5	0	0	coins inserted	pass	pass
sugar	3	25	5	5	0	0	sugar	pass	pass
small_cup	3	25	5	5	2	0	sugar	pass	pass
cancel	1	25	5	5	2	0	idle	pass	pass
set_price 50	1	50	5	5	2	0	idle	pass	pass
coin	1	50	5	5	2	25	idle	pass	pass

coin	2	50	5	5	0	0	coins inserted	pass	pass
sugar	3	50	5	5	0	0	sugar	pass	pass
large_cup	3	50	5	5	1	0	sugar	pass	pass
cancel	1	50	5	5	1	0	idle	pass	pass
coin	1	50	5	5	1	25	idle	pass	pass
coin	2	50	5	5	0	0	coins inserted	pass	pass
sugar	3	50	5	5	0	0	sugar	pass	pass
cancel	1	50	5	5	0	0	idle	pass	pass
dispose	6	50	5	5	0	0	shut down	pass	pass

Test#13:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 5	1	25	0	5	0	0	idle	pass	pass
coin	2	25	0	5	0	0	coins inserted	pass	pass
small_cup	2	25	0	5	2	0	coins inserted	pass	pass
sugar	3	25	0	5	2	0	sugar	pass	pass
tea	1	25	0	4	2	0	idle	pass	pass
coin	2	25	0	4	0	0	coins inserted	pass	pass
small_cup	2	25	0	4	2	0	coins inserted	pass	pass
sugar	3	25	0	4	2	0	sugar	pass	pass
tea	1	25	0	3	2	0	idle	pass	pass
insert_small_cups 5	1	25	0	8	2	0	idle	pass	pass
coin	2	25	0	8	0	0	coins inserted	pass	pass
sugar	3	25	0	8	0	0	sugar	pass	pass
small_cup	3	25	0	8	2	0	sugar	pass	pass
coin	3	25	0	8	2	0	sugar	pass	pass

tea	1	25	0	7	2	0	idle	pass	pass
insert_large_cups 5	1	25	5	7	2	0	idle	pass	pass
coin	2	25	5	7	0	0	coins inserted	pass	pass
sugar	3	25	5	7	0	0	sugar	pass	pass
small_cup	3	25	5	7	2	0	sugar	pass	pass
tea	1	25	5	6	2	0	idle	pass	pass
set_price 50	1	50	5	6	2	0	idle	pass	pass
coin	1	50	5	6	2	25	idle	pass	pass
coin	2	50	5	6	0	0	coins inserted	pass	pass
sugar	3	50	5	6	0	0	sugar	pass	pass
small_cup	3	50	5	6	2	0	sugar	pass	pass
tea	1	50	5	5	2	0	idle	pass	pass
coin	1	50	5	5	2	25	idle	pass	pass
coin	2	50	5	5	0	0	coins inserted	pass	pass
sugar	3	50	5	5	5	5	sugar	pass	pass
small_cup	3	50	5	5	2	0	sugar	pass	pass
tea	1	50	5	4	2	0	idle	pass	pass
dispose	6	50	5	4	2	0	shut down	pass	pass

Test#14:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 1	1	25	1	0	0	0	idle	pass	pass
coin	2	25	1	0	0	0	idle	pass	pass
large_cup	2	25	1	0	1	0	idle	pass	pass
tea	5	25	0	0	1	0	no large cups	pass	pass
insert_large_cups 1	1	25	1	0	1	0	idle	pass	pass

set_price 50	1	50	1	0	1	0	idle	pass	pass
insert_small_cups 1	1	50	1	1	1	0	idle	pass	pass
coin	1	50	1	1	1	25	idle	pass	pass
coin	2	50	1	1	0	0	coins inserted	pass	pass
large_cup	2	50	1	1	1	0	coins inserted	pass	pass
tea	5	50	0	1	1	0	no large cups	pass	pass
insert_large_cups 1	1	50	1	1	1	0	idle	pass	pass
coin	1	50	1	1	1	25	idle	pass	pass
coin	2	50	1	1	0	0	coins inserted	pass	pass
small_cup	2	50	1	1	2	0	coins inserted	pass	pass
tea	4	50	1	0	2	0	no small cups	pass	pass
insert_small_cups 1	1	50	1	1	2	0	idle	pass	pass
set_price 50	1	50	1	1	2	0	idle	pass	pass
coin	1	50	1	1	2	25	idle	pass	pass
coin	2	50	1	1	0	0	coins inserted	pass	pass
small_cup	2	50	1	1	2	0	coins inserted	pass	pass
tea	4	50	1	0	2	0	no small cups	pass	pass
insert_small_cups 1	1	50	1	1	2	0	idle	pass	pass
coin	1	50	1	1	2	25	idle	pass	pass
coin	2	50	1	1	0	0	coins inserted	pass	pass
small_cup	2	50	1	1	2	0	coins inserted	pass	pass
tea	4	50	1	0	2	0	no small cups	pass	pass
insert_small_cups 1	1	50	1	1	2	0	idle	pass	pass
dispose	6	50	1	1	2	0	shut down	pass	pass

Test#15:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass

set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	0	1	0	0	idle	pass	pass
coin	2	25	0	1	0	0	coins inserted	pass	pass
coin	2	25	0	1	0	0	coins inserted	pass	pass
cancel	1	25	0	1	0	0	idle	pass	pass
coin	2	25	0	1	0	0	coins inserted	pass	pass
small_cup	2	25	0	1	2	0	coins inserted	pass	pass
coin	2	25	0	1	2	0	coins inserted	pass	pass
tea	4	25	0	0	2	0	no small cups	pass	pass
insert_small_cups 5	1	25	0	5	2	0	idle	pass	pass
insert_large_cups 1	1	25	1	5	2	0	idle	pass	pass
coin	2	25	1	5	0	0	coins inserted	pass	pass
large_cup	2	25	1	5	1	0	coins inserted	pass	pass
coin	2	25	1	5	1	0	coins inserted	pass	pass
tea	5	25	0	5	1	0	no large cups	pass	pass
insert_large_cups 1	1	25	1	5	1	0	idle	pass	pass
coin	2	25	1	5	0	0	coins inserted	pass	pass
small_cup	2	25	1	5	2	0	coins inserted	pass	pass
cancel	1	25	1	5	2	0	idle	pass	pass
coin	2	25	1	5	0	0	coins inserted	pass	pass
small_cup	2	25	1	5	2	0	coins inserted	pass	pass
coin	2	25	1	5	2	0	coins inserted	pass	pass
tea	1	25	1	4	2	0	idle	pass	pass
set_price 50	1	50	1	4	2	0	idle	pass	pass
dispose	6	50	1	4	2	0	shut down	pass	pass

Test#16:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	

vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	0	1	0	0	idle	pass	pass
insert_large_cups 1	1	25	1	1	0	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
sugar	3	25	1	1	0	0	sugar	pass	pass
sugar	2	25	1	1	0	0	coins inserted	pass	pass
sugar	3	25	1	1	0	0	sugar	pass	pass
sugar	2	25	1	1	0	0	coins inserted	pass	pass
cancel	1	25	1	1	0	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
sugar	3	25	1	1	0	0	sugar	pass	pass
large_cup	3	25	1	1	1	0	sugar	pass	pass
sugar	2	25	1	1	1	0	coins inserted	pass	pass
tea	5	25	0	1	1	0	no large cups	pass	pass
insert_large_cups 5	1	25	5	1	1	0	idle	pass	pass
coin	2	25	5	1	0	0	coins inserted	pass	pass
sugar	3	25	5	1	0	0	sugar	pass	pass
small_cup	3	25	5	1	2	0	sugar	pass	pass
sugar	2	25	5	1	2	0	coins inserted	pass	pass
tea	4	25	5	0	2	0	no small cups	pass	pass
insert_small_cups 5	1	25	5	5	2	0	idle	pass	pass
coin	2	25	5	5	0	0	coins inserted	pass	pass
sugar	3	25	5	5	0	0	sugar	pass	pass
large_cup	3	25	5	5	1	0	sugar	pass	pass
sugar	2	25	5	5	1	0	coins inserted	pass	pass
tea	1	25	4	5	1	0	idle	pass	pass
coin	2	25	4	5	0	0	coins inserted	pass	pass
sugar	3	25	4	5	0	0	sugar	pass	pass

small_cup	3	25	4	5	2	0	sugar	pass	pass
sugar	2	25	4	5	2	0	coins inserted	pass	pass
tea	1	25	4	4	2	0	idle	pass	pass
set_price 50	1	50	4	4	2	0	idle	pass	pass
coin	1	50	4	4	2	25	idle	pass	pass
dispose	6	50	4	4	2	25	shut down	pass	pass

Test#17:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 1	1	25	1	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	1	1	0	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
sugar	3	25	1	1	0	0	sugar	pass	pass
small_cup	3	25	1	1	2	0	sugar	pass	pass
coin	3	25	1	1	2	0	sugar	pass	pass
tea	4	25	1	0	2	0	no small cups	pass	pass
insert_small_cups 5	1	25	1	5	2	0	idle	pass	pass
coin	2	25	1	5	0	0	coins inserted	pass	pass
sugar	3	25	1	5	0	0	sugar	pass	pass
large_cup	3	25	1	5	1	0	sugar	pass	pass
coin	3	25	1	5	1	0	sugar	pass	pass
tea	5	25	0	5	1	0	no large cups	pass	pass
insert_large_cups 1	1	25	1	5	1	0	idle	pass	pass
coin	2	25	1	5	0	0	coins inserted	pass	pass
sugar	3	25	1	5	0	0	sugar	pass	pass
small_cup	3	25	1	5	2	0	sugar	pass	pass

sugar	2	25	1	5	2	0	coins inserted	pass	pass
cancel	1	25	1	5	2	0	idle	pass	pass
coin	2	25	1	5	0	0	coins inserted	pass	pass
sugar	3	25	1	5	0	0	sugar	pass	pass
large_cup	3	25	1	5	1	0	sugar	pass	pass
sugar	2	25	1	5	1	0	coins inserted	pass	pass
cancel	1	25	1	5	1	0	idle	pass	pass
coin	2	25	1	5	0	0	coins inserted	pass	pass
large_cup	2	25	1	5	1	0	coins inserted	pass	pass
sugar	3	25	1	5	1	0	sugar	pass	pass
tea	5	25	0	5	1	0	no large cups	pass	pass
insert_large_cups 1	1	25	1	5	1	0	idle	pass	pass
dispose	6	25	1	5	1	0	shut down	pass	pass

Test#18:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
insert_large_cups 5	1	0	5	0	0	0	idle	pass	pass
set_price 25	1	25	5	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	5	1	0	0	idle	pass	pass
coin	2	25	5	1	0	0	idle	pass	pass
small_cup	2	25	5	1	2	0	coins inserted	pass	pass
sugar	3	25	5	1	2	0	sugar	pass	pass
tea	4	25	5	0	2	0	no small cups	pass	pass
insert_small_cups 1	1	25	5	1	2	0	idle	pass	pass
dispose	6	25	5	1	2	0	shut down	pass	pass

Test#19:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
insert_small_cups 5	1	0	0	5	0	0	idle	pass	pass
dispose	6	0	0	5	0	0	shut down	pass	pass

Test#20:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
dispose	6	0	0	0	0	0	shut down	pass	pass

Test#21:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
coin	1	0	0	0	0	0	idle	pass	pass
dispose	6	0	0	0	0	0	shut down	pass	pass

Test#22:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
small_cup	1	0	0	0	0	0	idle	pass	pass
large_cup	1	0	0	0	0	0	idle	pass	pass
sugar	1	0	0	0	0	0	idle	pass	pass
tea	1	0	0	0	0	0	idle	pass	pass
insert_large_cups 0	1	0	0	0	0	0	idle	pass	pass

insert_small_cups 0	1	0	0	0	0	0	idle	pass	pass
set_price 0	1	0	0	0	0	0	idle	pass	pass
cancel	1	0	0	0	0	0	idle	pass	pass
coin	1	0	0	0	0	0	idle	pass	pass
dispose	6	0	0	0	0	0	shut down	pass	pass

Test#23:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 5	1	25	5	0	0	0	idle	pass	pass
coin	2	25	5	0	0	0	coins inserted	pass	pass
tea	2	25	5	0	0	0	coins inserted	pass	pass
insert_large_cups 5	2	25	5	0	0	0	coins inserted	pass	pass
insert_small_cups 5	2	25	5	0	0	0	coins inserted	pass	pass
set_price 25	2	25	5	0	0	0	coins inserted	pass	pass
dispose	2	25	5	0	0	0	coins inserted	pass	pass
cancel	1	25	5	0	0	0	idle	pass	pass
dispose	6	25	5	0	0	0	shut down	pass	pass

Test#24:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 5	1	25	0	5	0	0	idle	pass	pass
coin	2	25	0	5	0	0	idle	pass	pass

sugar	3	25	0	5	0	0	sugar	pass	pass
tea	3	25	0	5	0	0	sugar	pass	pass
insert_large_cups 0	3	25	0	5	0	0	sugar	pass	pass
insert_small_cups 0	3	25	0	5	0	0	sugar	pass	pass
set_price 25	3	25	0	5	0	0	sugar	pass	pass
dispose	3	25	0	5	0	0	sugar	pass	pass
cancel	1	25	0	5	0	0	idle	pass	pass
dispose	6	25	5	0	0	0	shut down	pass	pass

Test#25:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	0	1	0	0	idle	pass	pass
coin	2	25	0	1	0	0	idle	pass	pass
small_cup	2	25	0	1	2	0	idle	pass	pass
tea	4	25	0	0	2	0	no small cups	pass	pass
small_cup	4	25	0	0	2	0	no small cups	pass	pass
large_cup	4	25	0	0	2	0	no small cups	pass	pass
sugar	4	25	0	0	2	0	no small cups	pass	pass
tea	4	25	0	0	2	0	no small cups	pass	pass
insert_large_cups 5	4	25	0	0	2	0	no small cups	pass	pass
insert_small_cups 0	4	25	0	0	2	0	no small cups	pass	pass
set_price 0	4	25	0	0	2	0	no small cups	pass	pass
cancel	4	25	0	0	2	0	no small cups	pass	pass
dispose	4	25	0	0	2	0	no small cups	pass	pass
insert_small_cups 5	4	25	0	5	2	0	idle	pass	pass
dispose	6	25	0	5	2	0	shut down	pass	pass

Test#26:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 1	1	25	1	0	0	0	idle	pass	pass
coin	2	25	1	0	0	0	coins inserted	pass	pass
large_cup	2	25	1	0	1	0	coins inserted	pass	pass
tea	5	25	0	0	1	0	no large cups	pass	pass
small_cup	5	25	0	0	1	0	no large cups	pass	pass
large_cup	5	25	0	0	1	0	no large cups	pass	pass
sugar	5	25	0	0	1	0	no large cups	pass	pass
tea	5	25	0	0	1	0	no large cups	pass	pass
insert_large_cups 0	5	25	0	0	1	0	no large cups	pass	pass
insert_small_cups 0	5	25	0	0	1	0	no large cups	pass	pass
set_price 25	5	25	0	0	1	0	no large cups	pass	pass
cancel	5	25	0	0	1	0	no large cups	pass	pass
dispose	5	25	0	0	1	0	no large cups	pass	pass
insert_large_cups 5	1	25	5	0	1	0	idle	pass	pass
dispose	6	25	5	0	1	0	shut down	pass	pass

Test#27:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
coin	2	25	0	0	0	0	coins inserted	pass	pass
tea	2	25	0	0	0	0	coins inserted	pass	pass
sugar	3	25	0	0	0	0	sugar	pass	pass

tea	3	25	0	0	0	0	sugar	pass	pass
sugar	2	25	0	0	0	0	coins inserted	pass	pass
cancel	1	25	0	0	0	0	idle	pass	pass
dispose	6	25	0	0	0	0	shut down	pass	pass

Test#28:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 1	1	25	1	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	1	1	0	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
tea	2	25	1	1	0	0	coins inserted	pass	pass
sugar	3	25	1	1	0	0	sugar	pass	pass
tea	3	25	1	1	0	0	sugar	pass	pass
cancel	1	25	1	1	0	0	idle	pass	pass
dispose	6	25	1	1	0	0	shut down	pass	pass

Test#29:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
coin	2	25	0	0	0	0	coins inserted	pass	pass
large_cup	2	25	0	0	1	0	coins inserted	pass	pass
tea	2	25	0	0	1	0	coins inserted	pass	pass
sugar	3	25	0	0	1	0	sugar	pass	pass

tea	3	25	0	0	1	0	sugar	pass	pass
cancel	1	25	0	0	1	0	idle	pass	pass
dispose	6	25	0	0	1	0	shut down	pass	pass

Test#30:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
coin	2	25	0	0	0	0	coins inserted	pass	pass
small_cup	2	25	0	0	2	0	coins inserted	pass	pass
tea	2	25	0	0	2	0	coins inserted	pass	pass
sugar	3	25	0	0	2	0	sugar	pass	pass
tea	3	25	0	0	2	0	sugar	pass	pass
cancel	1	25	0	0	2	0	idle	pass	pass
dispose	6	25	0	0	2	0	shut down	pass	pass

Test#31:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 5	1	25	5	0	0	0	idle	pass	pass
coin	2	25	5	0	0	0	coins inserted	pass	pass
tea	2	25	5	0	0	0	coins inserted	pass	pass
cancel	1	25	5	0	0	0	idle	pass	pass
insert_small_cups 5	1	25	5	5	0	0	idle	pass	pass
coin	2	25	5	5	0	0	coins inserted	pass	pass

tea	2	25	5	5	0	0	coins inserted	pass	pass
cancel	1	25	5	5	0	0	idle	pass	pass
dispose	6	25	5	5	0	0	shut down	pass	pass

Test#32:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
coin	2	25	0	0	0	0	coins inserted	pass	pass
large_cup	2	25	0	0	1	0	coins inserted	pass	pass
tea	2	25	0	0	1	0	coins inserted	pass	pass
small_cup	2	25	0	0	2	0	coins inserted	pass	pass
tea	2	25	0	0	2	0	coins inserted	pass	pass
cancel	1	25	0	0	2	0	idle	pass	pass
dispose	6	25	0	0	2	0	shut down	pass	pass

Test#33:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
insert_large_cups 1	1	25	1	0	0	0	idle	pass	pass
insert_small_cups 1	1	25	1	1	0	0	idle	pass	pass
coin	2	25	1	1	0	0	coins inserted	pass	pass
tea	2	25	1	1	0	0	coins inserted	pass	pass
sugar	3	25	1	1	0	0	sugar	pass	pass
tea	3	25	1	1	0	0	sugar	pass	pass

cancel	1	25	1	1	0	0	idle	pass	pass
dispose	6	25	1	1	0	0	shut down	pass	pass

Test#34:

Operation Invoked	Expected values after invoking the method								Actual Result
	x	price	k	k1	s	t	state	result	
vending_machine	1	0	0	0	0	0	idle	pass	pass
set_price 25	1	25	0	0	0	0	idle	pass	pass
dispose	6	25	0	0	0	0	shut down	pass	pass
coin	6	25	0	0	0	0	shut down	pass	pass
dispose	6	25	0	0	0	0	shut down	pass	pass

7. Conclusions:

As I started to work on this project, my first step was to understand the EFSM model properly. Once I understood the EFSM model, it became a lot easier to do the transition pairing testing. The second step was to do the multiple condition testing. Multiple condition testing was very time consuming especially the method tea() had a lot of possible multiple conditions and so I had to concentrate pretty well while writing the test cases. Lastly, I performed the Ghost Transition Testing which was a lot easier than the previous two methods.

The most important phase was the implementation of the testing environment, which is the test driver in this project. First of all I looked at the code and started implementing the test driver. It did not take a lot of time to implement the driver, but to test each and every test case manually in the test driver by executing every method/transition one by one was a time consuming task. Moreover, I had to concentrate pretty hard while testing because even if one mistake was made, I had to start the test all over again. But to sum up, the test driver is a pretty nice way to execute the test cases because it can catch errors which were made when testing manually. I compared each and every test case with the actual result and expected result with the help of the test driver.

Some activities related to class testing could be automated like when we had to run/execute each and every test case manually through the driver, it was a tedious task. But if we could automate/code the driver such that it takes the TS.txt file as input and gives us pass/fail result for every test case in the test suite, it would reduce a significant amount of time compared to manual testing. Moreover, automation can be done in identifying and checking all the executable and non-executable multiple conditions which would again reduce a lot of time when compared to manually testing the multiple conditions.

8. Source Code:

8.1. VendingMachine Class:

```
package staproject;
/**
 *
 * @author Mohammed
 */
public class VendingMachine {
    private int x;
    private int price;
    private int k;
    private int k1;
    private int t;
    private int s;

    public VendingMachine()
    {
        k1 = 0;
        k = 0;
        t = 0;
        price = 0;
        x = 1;
    }
    public final int coin()
    {
        if (x == 1)
        {
```

```
            if ((t + 25 >= price) && (price > 0))
            {
                s = 0;
                t = 0;
                x = 2;
                return 1;
            }
            else if (t + 25 < price)
            {
                t = t + 25;
                return 1;
            }
        }
        else if ((x > 1) && (x < 6))
        {
            System.out.print("RETURN COIN");
            System.out.print("\n");
            return 1;
        }
        return 0;
    }
    public final int small_cup()
    {
        if ((x == 2) || (x == 3))
        {
            s = 2;
            return 1;
```

```
        }
        return 0;
    }
    public final int large_cup()
    {
        if ((x == 2) || (x == 3))
        {
            s = 1;
            return 1;
        }
        return 0;
    }
    public final int sugar()
    {
        if ((x == 2) || (x == 3))
        {
            if (x == 2)
            {
                x = 3;
            }
            else
            {
                x = 2;
            }
            return 1;
        }
        return 0;
```

```
}
public final int tea()
{
    if ((x == 2) || (x == 3))
    {
        if ((x == 2) && (k1 > 1) && (s == 2))
        {
            System.out.print("DISPOSE SMALL CUP OF TEA");
            System.out.print("\n");
            k1 = k1 - 1;
            x = 1;
            return 1;
        }
        else if ((x == 2) && (k > 1) && (s == 1))
        {
            System.out.print("DISPOSE LARGE CUP OF TEA");
            System.out.print("\n");
            k = k - 1;
            x = 1;
            return 1;
        }
        else if ((x == 2) && (k == 1) && (s == 1))
        {
            System.out.print("DISPOSE LARGE CUP OF TEA");
            System.out.print("\n");
            k = k - 1;
            x = 5;
```

```
        return 1;
}
else if ((x == 2) && (k1 == 1) && (s == 2))
{
        System.out.print("DISPOSE SMALL CUP OF TEA");
        System.out.print("\n");
        k1 = k1 - 1;
        x = 4;
        return 1;
}
else if ((x == 3) && (k1 == 1) && (s == 2))
{
        System.out.print("DISPOSE SMALL CUP OF TEA WITH SUGAR");
        System.out.print("\n");
        k1 = k1 - 1;
        x = 4;
        return 1;
}
else if ((x == 3) && (k == 1) && (s == 1))
{
        System.out.print("DISPOSE LARGE CUP OF TEA WITH SUGAR");
        System.out.print("\n");
        k = k - 1;
        x = 5;
        return 1;
}
if ((x == 3) && (k1 > 1) && (s == 2))
```

```
            {
                System.out.print("DISPOSE SMALL CUP OF TEA WITH SUGAR");
                System.out.print("\n");
                k1 = k1 - 1;
                x = 1;
                return 1;
            }
            else if ((x == 3) && (k > 1) && (s == 1))
            {
                System.out.print("DISPOSE LARGE CUP OF TEA WITH SUGAR");
                System.out.print("\n");
                k = k - 1;
                x = 1;
                return 1;
            }
            return 0;
        }
        return 0;
    }
    public final int insert_large_cups(int n)
    {
        if ((x == 1) && (n > 0))
        {
            k = k + n;
            return 1;
        }
        else if ((x == 5) && (n > 0))
```

```
        {
            k = n;
            x = 1;
            return 1;
        }
        return 0;
    }
    public final int insert_small_cups(int n)
    {
        if ((x == 1) && (n > 0))
        {
            k1 = k1 + n;
            return 1;
        }
        else if ((x == 4) && (n > 0))
        {
            k1 = n;
            x = 1;
            return 1;
        }
        return 0;
    }
    public final int set_price(int p)
    {
        if ((x == 1) && (p > 0))
        {
            price = p;
```

```
            return 1;
        }
        return 0;
    }
    public final int cancel()
    {
        if ((x == 2) || (x == 3))
        {
            System.out.print("RETURN COINS");
            System.out.print("\n");
            x = 1;
            return 1;
        }
        return 0;
    }
    public final int dispose()
    {
        if ((x == 1))
        {
            System.out.print("SHUT DOWN");
            System.out.print("\n");
            x = 6;
            return 1;
        }
        return 0;
    }
```

```
//Testing oriented methods...

   //Shows the current state of the VendingMachine Class
   void show_state()
   {
      System.out.println("");
      if(x==1){
            System.out.println("current state: idle state");
      }else if(x==2){
            System.out.println("current state: coins inserted state");
      }else if(x==3){
            System.out.println("current state: sugar state");
      }else if(x==4){
            System.out.println("current state: no small_cups state");
      }else if(x==5){
            System.out.println("current state: no large_cups state");
      }else if(x==6){
            System.out.println("current state: Shut Down state");
            System.out.println("Start the test driver again to execute another test case !");
      }
      System.out.println("");
   }

   //Shows all the private variable values of the class VendingMachine
   void show_all_values(){

      System.out.println("x="+x);
```

```
    System.out.println("price="+price);
    System.out.println("k="+k);
    System.out.println("k1="+k1);
    System.out.println("t="+t);
    System.out.println("s="+s);
}

//Shows variable x value
void show_x(){
    System.out.println("x="+x);
}

//Shows variable price value
void show_price(){
    System.out.println("price="+price);
}

//Shows variable k value
void show_k(){
    System.out.println("k="+k);
}

//Shows variable k1 value
void show_k1(){
    System.out.println("k1="+k1);
}
```

```
    //Shows variable t value
    void show_t(){
        System.out.println("t="+t);
    }

    //Shows variable s value
    void show_s(){
        System.out.println("s="+s);
    }

}
```

8.2. Test Driver (TestDriverVendingMachine) Class:

```
package staproject;
import java.io.*;
import java.util.Scanner;
/**
 *
 * @author Mohammed
 */
public class TestDriverVendingMachine {

   public static void main(String[] args) {

      int userOption=0;
      int return_value=-1;

      //Create an object of VendingMachine class. It's constructor will be invoked
      VendingMachine obj=new VendingMachine();
      //Get the key pressed from the keyboard
      Scanner in = new Scanner(System.in);

      System.out.println("Driver for the Vending Machine");

      //If the user presses 11, then driver will close, otherwise it will iterate in this loop
      while(userOption!=11){

            System.out.println("-----------------------------------");
                  System.out.println("Please choose from below options:");
```

```
        System.out.println("1. coin()");
        System.out.println("2. small_cup()");
        System.out.println("3. large_cup()");
        System.out.println("4. sugar()");
        System.out.println("5. tea()");
        System.out.println("6. insert_large_cups(int n)");
        System.out.println("7. insert_small_cups(int n)");
        System.out.println("8. set_price(int p)");
        System.out.println("9. cancel()");
        System.out.println("10. dispose()");

System.out.println("-----------------------------------");
        System.out.println("11. Quit");

System.out.println("-----------------------------------");

        System.out.println("Testing Oriented Methods:");
        System.out.println("12. show_state()");
        System.out.println("13. show_all_values()");
        System.out.println("14. show_x()");
System.out.println("15. show_price()");
System.out.println("16. show_k()");
        System.out.println("17. show_k1()");
        System.out.println("18. show_t()");
        System.out.println("19. show_s()");

System.out.println("-----------------------------------");
```

```
System.out.println("");
        System.out.println("Enter number: ");

        userOption=in.nextInt();

//This switch is used to display the user selected method on the screen
        switch(userOption){
        case 1:
                System.out.println("coin()");
                return_value=obj.coin();
                System.out.println("The return value="+return_value);
                break;
        case 2:
                System.out.println("small_cup()");
                return_value=obj.small_cup();
                System.out.println("The return value="+return_value);
                break;
        case 3:
                System.out.println("large_cup()");
                return_value=obj.large_cup();
                System.out.println("The return value="+return_value);
                break;
        case 4:
                System.out.println("sugar()");
                return_value=obj.sugar();
                System.out.println("The return value="+return_value);
                break;
```

```
case 5:
        System.out.println("tea()");
        return_value=obj.tea();
        System.out.println("The return value="+return_value);
        break;
case 6:
        int n;
        System.out.println("insert_large_cups(int n)");
        System.out.println("Enter value of n:");
        n=in.nextInt();
        return_value=obj.insert_large_cups(n);
        System.out.println("The return value="+return_value);
        break;
case 7:
        int n1;
        System.out.println("insert_small_cups(int n)");
        System.out.println("Enter value of n:");
        n1=in.nextInt();
        return_value=obj.insert_small_cups(n1);
        System.out.println("The return value="+return_value);
        break;
case 8:
        int p;
        System.out.println("set_price(int p)");
        System.out.println("Enter value of p:");
        p=in.nextInt();
        return_value=obj.set_price(p);
```

```
            System.out.println("The return value="+return_value);
            break;
    case 9:
            System.out.println("cancel()");
            return_value=obj.cancel();
            System.out.println("The return value="+return_value);
            break;
    case 10:
            System.out.println("dispose()");
            return_value=obj.dispose();
            System.out.println("The return value="+return_value);
            break;

    case 12:
System.out.println("show_state()");
            obj.show_state();
            break;
    case 13:
System.out.println("show_all_values()");
            obj.show_all_values();
            break;
    case 14:
System.out.println("show_x()");
            obj.show_x();
            break;
    case 15:
System.out.println("show_price()");
```

```
                obj.show_price();
                break;
        case 16:
            System.out.println("show_k()");
                obj.show_k();
                break;
          case 17:
            System.out.println("show_k1()");
                obj.show_k1();
                break;
        case 18:
            System.out.println("show_t()");
                obj.show_t();
                break;
          case 19:
            System.out.println("show_s()");
                obj.show_s();
                break;
          default:
                if(userOption!=11)
            {
              System.out.println("Please choose only from the given options!!");
              System.out.println("Else Enter 11 to quit");
                }

            break;
              }
```

```
                System.out.println("");

            }

            System.out.println("Test Driver is Stopped");
            System.exit(0);
        }

}
```