1. Model Based Testing (Transition Pairs Testing):

state	In coming transition	Outgoing transition
idle	T1, T2, T3, T4, T6, T8, T9, T10, T11, T12, T13, T14, T15	T2, T3, T4, T5, T6, T7
coins inserted	T7, T19, T20, T21, T23	T10, T11, T12, T19, T20, T21, T22, T24, T25
sugar	T16, T17, T18, T22	T13, T14, T15, T16, T17, T18, T23, T26, T27
no large_cups	T24, T26, T29	T8, T29
no small_cups	T25, T27, T28	T9, T28

For state: idle

Transition Pairs	Covered?	Transition Pairs	Covered?	Transition Pairs	Covered?
T1, T2	Yes	T8, T3	Yes	T13, T4	Yes
T1, T3	Yes	T8, T4	Yes	T13, T5	Yes
T1, T4	Yes	T8, T5	Yes	T13, T6	Yes
T1, T5	Yes	T8, T6	Yes	T13, T7	Yes
T1, T6	No test	T8, T7	Yes	T14, T2	Yes
T1, T7	No test	T9, T2	Yes	T14, T3	Yes
T2, T2	Yes	T9, T3	Yes	T14, T4	Yes
T2, T3	Yes	T9, T4	Yes	T14, T5	Yes
T2, T4	Yes	T9, T5	Yes	T14, T6	Yes
T2, T5	Yes	T9, T6	Yes	T14, T7	Yes
T2, T6	Yes	T9, T7	Yes	T15, T2	Yes
T2, T7	Yes	T10, T2	Yes	T15, T3	Yes
T3, T2	Yes	T10, T3	Yes	T15, T4	Yes
T3, T3	Yes	T10, T4	Yes	T15, T5	Yes
T3, T4	Yes	T10, T5	Yes	T15, T6	Yes
T3, T5	Yes	T10, T6	Yes	T15, T7	Yes
T3, T6	Yes	T10, T7	Yes		
T3, T7	Yes	T11, T2	Yes		
T4, T2	Yes	T11, T3	Yes		
T4, T3	Yes	T11, T4	Yes		
T4,T4	Yes	T11, T5	Yes		
T4, T5	Yes	T11, T6	Yes		
T4, T6	Yes	T11, T7	Yes		
T4, T7	Yes	T12, T2	Yes		
T6, T2	Yes	T12, T3	Yes		
T6, T3	Yes	T12, T4	Yes		
T6, T4	Yes	T12, T5	Yes		
T6, T5	Yes	T12, T6	Yes		
T6, T6	Yes	T12, T7	Yes		
T6, T7	Yes	T13, T2	Yes		
T8, T2	Yes	T13, T3	Yes		

Reason for non-executable pair (T1, T6): T1 is by default called when the system starts since it's the transition of the constructor. At this time, t=0 and price =0. For transition T6, "(t + 25) < price" should satisfy.

But, when the coin() is called at this time, t = 0 and price = 0, so 0 + 25 >0. Thus, T6 cannot take place. Hence, (T1, T6) is non-executable.

Reason for non-executable pair (T1, T7): T1 is by default called when the system starts since it's the transition of the constructor. At this time, t=0 and price =0. For transition T7, "(t + 25) >= price && price > 0" should satisfy.

But, when the coin() is called at this time, t = 0 and price = 0, so 0 + 25 >0. But, price =0. Thus, T7 cannot take place. Hence, (T1, T7) is non-executable.

For state: coins inserted

Transition Pairs	Covered?	Transition Pairs	Covered?
T7, T10	Yes	T21, T10	Yes
T7, T11	No test	T21, T11	Yes
T7, T12	No test	T21, T12	No test
T7, T19	Yes	T21, T19	Yes
T7, T20	Yes	T21, T20	Yes
T7, T21	Yes	T21, T21	Yes
T7, T22	Yes	T21, T22	Yes
T7, T24	No test	T21, T24	No test
T7, T25	No test	T21, T25	Yes
T19. T10	Yes	T23, T10	Yes
T19, T11	No test	T23, T11	Yes
T19, T12	Yes	T23, T12	Yes
T19, T19	Yes	T23, T19	Yes
T19, T20	Yes	T23, T20	Yes
T19, T21	Yes	T23, T21	Yes
T19, T22	Yes	T23, T22	Yes
T19, T24	Yes	T23, T24	Yes
T19, T25	No test	T23, T25	Yes
T20, T10	Yes		
T20, T11	Yes		
T20, T12	Yes		
T20, T19	Yes		
T20, T20	Yes		
T20, T21	Yes		
T20, T22	Yes		
T20, T24	Yes		
T20, T25	Yes		

Reason for non-executable pair (T7, T11): When T7 occurs, s=0 and t=0. For T11 to occur, (k1>1 && s==2), but s=0. Hence (T7, T11) is non-executable.

Reason for non-executable pair (T7, T12): When T7 occurs, s=0 and t=0. For T12 to occur, (k>1 && s==1), but s=0. Hence (T7, T12) is non-executable.

Reason for non-executable pair (T7, T24): When T7 occurs, s=0 and t=0. For T24 to occur, (k ==1 && s==1), but s=0. Hence (T7, T24) is non-executable.

Reason for non-executable pair (T7, T25): When T7 occurs, s=0 and t=0. For T25 to occur, (k1==1 && s==2), but s=0. Hence (T7, T25) is non-executable.

Reason for non-executable pair (T19, T11): When T19 occurs, s=1. For T11 to occur, (k1>1 && s==2), but s=1. Hence (T19, T11) is non-executable.

Reason for non-executable pair (T19, T25): When T19 occurs, s=1. For T25 to occur, (k1==1 && s==2), but s=1. Hence (T19, T25) is non-executable.

Reason for non-executable pair (T21, T12): When T21 occurs, s=2. For T12 to occur, (k>1 && s==1), but s=2. Hence (T21, T12) is non-executable.

Reason for non-executable pair (T21, T24): When T21 occurs, s=2. For T24 to occur, (k==1 && s==1), but s=2. Hence (T21, T24) is non-executable.

For state: sugar

Transition Pairs	Covered?	Transition Pairs	Covered?
T16, T13	Yes	T18, T13	Yes
T16, T14	Yes	T18, T14	Yes
T16, T15	Yes	T18, T15	No test
T16, T16	Yes	T18, T16	Yes
T16, T17	Yes	T18, T17	Yes
T16, T18	Yes	T18, T18	Yes
T16, T23	Yes	T18, T23	Yes
T16, T26	Yes	T18, T26	Yes
T16, T27	Yes	T18, T27	No test
T17, T13	No test	T22, T13	Yes
T17, T14	Yes	T22, T14	Yes
T17, T15	Yes	T22, T15	Yes
T17, T16	Yes	T22, T16	Yes
T17, T17	Yes	T22, T17	Yes
T17, T18	Yes	T22, T18	Yes
T17, T23	Yes	T22, T23	Yes
T17, T26	No test	T22, T26	Yes
T17, T27	Yes	T22, T27	Yes

Reason for non-executable pair (T17, T13): When T17 occurs, s=2. For T13 to occur, (k>1 && s==1), but s=2. Hence (T17, T13) is non-executable.

Reason for non-executable pair (T17, T26): When T17 occurs, s=2. For T26 to occur, (k==1 && s==1), but s=2. Hence (T17, T26) is non-executable.

Reason for non-executable pair (T18, T15): When T18 occurs, s=1. For T15 to occur, (k1>1 && s==2), but s=1. Hence (T18, T15) is non-executable.

Reason for non-executable pair (T18, T27): When T18 occurs, s=1. For T27 to occur, (k1==1 && s==2), but s=1. Hence (T18, T27) is non-executable.

For state: no large_cups

Transition Pairs	Covered?
T24, T8	Yes
T24, T29	Yes
T26, T8	Yes
T26, T29	Yes
T29, T8	Yes
T29, T29	Yes

For state: no small_cups

Transition Pairs	Covered?
T25, T9	Yes
T25, T28	Yes
T27, T9	Yes
T27, T28	Yes
T28, T9	Yes
T28, T28	Yes

Test Case	Transition Traversed	Transition Pair Covered
Test#1	T1, T4, T2, T6, T6, T6, T7, T19, T24, T8, T5	(T1, T4), (T4, T2), (T2, T6), (T6, T6), (T6, T6), (T6, T7), (T7, T19), (T19, T24), (T24, T8), (T8, T5)
Test#2	T1, T2, T4, T6, T5	(T1, T2), (T2, T4), (T4, T6), (T6, T5)
Test#3	T1, T4, T3, T6, T6, T7, T21, T21, T11, T6, T6, T7, T21, T10, T5	(T1, T4), (T4, T3), (T3, T6), (T6, T6), (T6, T7), (T7, T21), (T21, T21), (T21, T11), (T11, T6), (T6, T6), (T6, T7), (T7, T21), (T21, T10), (T10, T5)
Test#4	T1, T3, T4, T2, T6, T7, T19, T12, T6, T7, T21, T25, T28, T9, T5	(T1, T3), (T3, T4), (T4, T2), (T2, T6), (T6, T7), (T7, T19), (T19, T12), (T12, T6), (T6, T7), (T7, T21), (T21, T25), (T25, T28), (T28, T9), (T9, T5)
Test#5	T1, T5	(T1, T5)
Test#6	T1, T2, T2, T3, T3, T4, T7, T10, T7, T20, T10, T7, T22, T16, T17, T23, T11, T5	(T1, T2), (T2, T2), (T2, T3), (T3, T3), (T3, T4), (T4, T7), (T7, T10), (T10, T7), (T7, T20), (T20, T10), (T10, T7), (T7, T22), (T22, T16), (T16, T17), (T17, T23), (T23, T11), (T11, T5)
Test#7	T1, T3, T2, T5	(T1, T3), (T3, T2), (T2, T5)
Test#8	T1, T3, T5	(T1,T3), (T3, T5)
Test#9	T1, T4, T4, T2, T7, T20, T22, T23, T22, T18, T26, T29, T29, T8, T4, T5	(T1, T4), (T4, T4), (T4, T2), (T2, T7), (T7, T20), (T20, T22), (T22, T23), (T23, T22), (T22, T18), (T18, T26), (T26, T29), (T29, T29), (T29, T8), (T8, T4), (T4, T5)
Test#10	T1, T4, T3, T7, T21, T22, T17, T27, T28, T28, T9, T7, T22, T17, T15, T5	(T1, T4), (T4, T3), (T3, T7), (T7, T21), (T21, T22), (T22, T17), (T17, T27), (T27, T28), (T28, T28), (T28, T9), (T9, T7), (T7, T22), (T22, T17), (T17, T15), (T15, T5)
Test#11	T1, T4, T6, T2, T6, T3, T6, T4, T6, T7, T20, T20, T19, T19, T20, T12, T4, T7, T20, T21, T20, T25, T9, T4, T6, T7, T19, T20, T24, T29, T8, T3, T5	(T1, T4), (T4, T6), (T6, T2), (T2, T6), (T6, T3), (T3, T6), (T6, T4), (T4, T6), (T6, T7), (T7, T20), (T20, T20), (T20, T19), (T19, T19), (T19, T20), (T20, T12), (T12, T4), (T4, T7), (T7, T20, (T20, T21), (T21, T20), (T20, T25), (T25, T9), (T9, T4), (T4, T6), (T6, T7), (T7, T19), (T19, T20), (T20, T24), (T24, T29), (T29, T8), (T8, T3), (T3, T5)
Test#12	T1, T4, T3, T7, T22, T16, T16, T17, T27, T9, T2, T7, T22, T18, T16, T18, T26, T8, T2, T5	(T1, T4), (T4, T3), (T3, T7), (T7, T22), (T22, T16), (T16, T16), (T16, T17), (T17, T27), (T27, T9), (T9, T2), (T2, T7), (T7, T22), (T22, T18), (T18, T16), (T16, T18), (T18, T26), (T26, T8), (T8, T2), (T2, T5)
Test#13	T1, T4, T3, T7, T19, T10, T7, T19, T21, T11, T7, T19, T22, T23, T20, T22, T23, T21, T19, T21, T20, T11, T5	(T1, T4), (T4, T3), (T3, T7), (T7, T19), (T19, T10), (T10, T7), (T7, T19), (T19, T21), (T21, T11), (T11, T7), (T7, T19), (T19, T22), (T22, T23), (T23, T20), (T20, T22), (T22, T23), (T23, T21), (T21, T19), (T19, T21), (T21, T20), (T20, T11), (T11, T5)
Test#14	T1, T4, T2, T7, T22, T23, T10, T7, T22, T23, T19, T22, T23, T12, T5	(T1, T4), (T4, T2), (T2, T7), (T7, T22), (T22, T23), (T23, T10), (T10, T7), (T7, T22), (T22, T23), (T23, T19), (T19, T22), (T22, T23), (T23, T12), (T12, T5)
Test#15	T1, T4, T2, T6, T7, T22, T14, T6, T7, T22, T18, T16, T13, T3, T6, T7, T22, T18, T16, T26, T8, T6, T7, T22, T17, T17, T16, T15, T3, T5	(T1, T4), (T4, T2), (T2, T6), (T6, T7), (T7, T22), (T22, T14), (T14, T6), (T6, T7), (T7, T22), (T22, T18), (T18, T16), (T16, T13), (T13, T3), (T3, T6), (T6, T7), (T7, T22), (T22, T18), (T18, T16), (T16, T26), (T26, T8), (T8, T6), (T6, T7), (T7, T22), (T22, T17), (T17, T17), (T17, T16), (T16, T15), (T15, T3), (T3, T5)

Test#16	T1, T4, T2, T7, T19, T22, T23, T24, T8, T7, T22, T18, T14, T3, T7, T21, T22, T23, T25, T9, T3, T5	(T1, T4), (T4, T2), (T2, T7), (T7, T19), (T19, T22), (T22, T23), (T23, T24), (T24, T8), (T8, T7), (T7, T22), (T22, T18), (T18, T14), (T14, T3), (T3, T7), (T7, T21), (T21, T22), (T22, T23), (T23, T25), (T25, T9), (T9, T3), (T3, T5)
Test#17	T1, T4, T3, T6, T7, T21, T22, T15, T2, T6, T7, T19, T22, T13, T6, T7, T21, T22, T27, T9, T6, T7, T19, T22, T26, T8, T5	(T1, T4), (T4, T3), (T3, T6), (T6, T7), (T7, T21), (T21, T22), (T22, T15), (T15, T2), (T2, T6), (T6, T7), (T7, T19), (T19, T22), (T22, T13), (T13, T6), (T6, T7), (T7, T21), (T21, T22), (T22, T27), (T27, T9), (T9, T6), (T6, T7), (T7, T19), (T19, T22), (T22, T26), (T26, T8), (T8, T5)
Test#18	T1, T4, T3, T2, T7, T22, T17, T18, T17, T16, T14, T7, T22, T17, T16, T27, T9, T7, T22, T18, T18, T23, T22, T18, T13, T2, T5	(T1, T4), (T4, T3), (T3, T2), (T2, T7), (T7, T22), (T22, T17), (T17, T18), (T18, T17), (T17, T16), (T16, T14), (T14, T7), (T7, T22), (T22, T17), (T17, T16), (T16, T27), (T27, T9), (T9, T7), (T7, T22), (T22, T18), (T18, T18), (T18, T23), (T23, T22), (T22, T18), (T18, T13), (T13, T2), (T2, T5)
Test#19	T1, T4, T2, T7, T22, T16, T23, T22, T17, T14, T4, T7, T10, T4, T6, T7, T10, T6, T7, T10, T3, T6, T7, T10, T2, T6, T7, T10, T5	(T1, T4), (T4, T2), (T2, T7), (T7, T22), (T22, T16), (T16, T23), (T23, T22), (T22, T17), (T17, T14), (T14, T4), (T4, T7), (T7, T10), (T10, T4), (T4, T6), (T6, T7), (T7, T10), (T10, T6), (T6, T7), (T7, T10), (T10, T3), (T3, T6), (T6, T7), (T7, T10), (T10, T2), (T2, T6), (T6, T7), (T7, T10), (T10, T5)
Test#20	T1, T3, T7, T21, T11, T2, T7, T21, T11, T3, T7, T21, T11, T4, T7, T21, T11, T7, T22, T17, T15, T7, T22, T17, T15, T4, T6, T7, T22, T17, T15, T6, T7, T10, T5	(T1, T3), (T3, T7), (T7, T21), (T21, T11), (T11, T2), (T2, T7), (T7, T21), (T21, T11), (T11, T3), (T3, T7), (T7, T21), (T21, T11), (T11, T4), (T4, T7), (T7, T21), (T21, T11), (T11, T7), (T7, T22), (T22, T17), (T17, T15), (T15, T7), (T7, T22), (T22, T17), (T17, T15), (T15, T4), (T4, T6), (T6, T7), (T7, T22), (T22, T17), (T17, T15), (T15, T6), (T6, T7), (T7, T10), (T10, T5)
Test#21	T1, T4, T2, T7, T19, T12, T2, T7, T19, T12, T3, T7, T19, T12, T7, T22, T18, T13, T7, T22, T18, T13, T4, T6, T7, T19, T12, T5	(T1, T4), (T4, T2), (T2, T7), (T7, T19), (T19, T12), (T12, T2), (T2, T7), (T7, T19), (T19, T12), (T12, T3), (T3, T7), (T7, T19), (T19, T12), (T12, T7), (T7, T22), (T22, T18), (T18, T13), (T13, T7), (T7, T22), (T22, T18), (T18, T13), (T13, T4), (T4, T6), (T6, T7), (T7, T19), (T19, T12), (T12, T5)
Test#22	T1, T4, T3, T6, T6, T6, T7, T22, T14, T2, T6, T6, T6, T7, T22, T14, T5	(T1, T4), (T4, T3), (T3, T6), (T6, T6), (T6, T6), (T6, T7), (T7, T22), (T22, T14), (T14, T2), (T2, T6), (T6, T6), (T6, T6), (T6, T7), (T7, T22), (T22, T14), (T14, T5)

2. Default (Ghost) Transition Testing:

State	Default Transitions	Test
idle	coin() set_price(p<=0) insert_large_cups(n<=0) insert_small_cups(n<=0) small_cup() large_cup() tea() sugar() cancel()	Test#32
no_large_cups	set_price(p) *(Any value of 'p' will result in default transition)* small_cup() large_cup() tea() sugar() cancel() dispose() insert_small_cups(n) *(Any value of 'n' will result in default transition)* insert_large_cups(n<=0)	Test#33
no_small_cups	set_price(p) *(Any value of 'p' will result in default transition)* small_cup() large_cup() tea() sugar() cancel() dispose() insert_large_cups(n) *(Any value of 'n' will result in default transition)* insert_small_cups(n<=0)	Test#34
inserted_coin	set_price(p) *(Any value of 'p' will result in default transition)* dispose() insert_small_cups(n) *(Any value of 'n' will result in default transition)* insert_large_cups(n) *(Any value of 'n' will result in default transition)* tea() *// (k<1 && s Є [1-(1,2)]) OR (k1<1 && s Є [1-(1,2)])*	Test#35
sugar	set_price(p) *(Any value of 'p' will result in default transition)* dispose() insert_small_cups(n) *(Any value of 'n' will result in default transition)* insert_large_cups(n) *(Any value of 'n' will result in default transition)* tea() *// (k<1 && s Є [1-(1,2)]) OR (k1<1 && s Є [1-(1,2)])*	Test#35

3. Multiple Condition Testing:

Method: coin()

For Condition: (x==1)

T	Test #1
F	Test #6

For Condition: (t+25 >= price) && (price > 0)
c1: t+25 >= price
c2: price > 0

c1	c2		Reason for non-executable
T	T	Test#10	
T	F	Test#24	
F	T	Test#1	
F	F	Non- executable	Initially, "price=0" and "t = 0" when the vending machine starts. So, when at "x=1" when vending machine starts and coin() is called, price = 0 and t = 0. So, (0+25) >= 0. So, c1 is true. Also, while returning back to "x=1" from other states, t= 0 and price >0. Moreover, set_price() allows only positive values of price greater than 0. Hence, the value of t>= 0 and value of price >=0, when at x=1.

For Condition: (t+25 < price)

		Reason for non-executable
T	Test#1	
F	Non- executable	We can reach "else if" in the code only if "if" condition is false. ! [(t+25 >= price) && (price > 0)] = !(t+25 >= price) || !(price > 0) = t + 25 < price || price <=0 Thus, (t + 25 < price) will be executed if "if" condition is false.

For Condition: (x > 1) && (x < 6)

c1: x > 1

c2: x < 6

c1	c2		Reason for non-executable
T	T	Test#6	
T	F	Test#5	
F	T	Non- executable	!(x > 1) = x<=1. The value of "x" when the system starts is 1. Thus, the minimum value of x is 1. Hence, x=1. Thus, "if" condition will be executed and code will not reach (x > 1) && (x < 6). We can reach "else if" in the code only if "if" condition is false. Therefore, we can say that (x<=1 && x < 6) is not possible.
F	F	Non- executable	!(x > 1) = x<=1. The value of "x" when the system starts is 1. Thus, the minimum value of x is 1. Hence, x=1. Thus, "if" condition will be executed and code will not reach (x > 1) && (x < 6). We can reach "else if" in the code only if "if" condition is false. Therefore, we can say that (F,F) is not possible.

Method: small_cup()

For Condition: (x == 2) || (x == 3)

c1: x == 2

c2: x == 3

c1	c2		Reason for non-executable
T	T	Non- executable	value of "x" cannot be "2" and "3" simultaneously
T	F	Test#3	
F	T	Test#6	
F	F	Test#24	

Method: large_cup()

For Condition: (x == 2) || (x == 3)

c1: x == 2

c2: x == 3

c1	c2		Reason for non-executable
T	T	Non- executable	value of "x" cannot be "2" and "3" simultaneously
T	F	Test#1	
F	T	Test#9	
F	F	Test#24	

Method: sugar()

For Condition: (x == 2) || (x == 3)

c1: x == 2

c2: x == 3

c1	c2		Reason for non-executable
T	T	Non- executable	value of “x” cannot be “2” and “3” simultaneously
T	F	Test#6	
F	T	Test#6	
F	F	Test#24	

For Condition: (x == 2)

T	Test#6
F	Test#6

Method: tea()

For Condition: (x == 2) || (x == 3)

c1: x == 2

c2: x == 3

c1	c2		Reason for non-executable
T	T	Non- executable	value of “x” cannot be “2” and “3” simultaneously
T	F	Test#1	
F	T	Test#9	
F	F	Test#24	

For Condition: (x == 2) && (k1 > 1) && (s == 2)

c1: x == 2

c2: k1 > 1

c3: s == 2

c1	c2	c3	
T	T	T	Test#3
T	T	F	Test#25
T	F	T	Test#26
T	F	F	Test#26
F	T	T	Test#10
F	T	F	Test#27
F	F	T	Test#28
F	F	F	Test#28

For Condition: (x == 2) && (k > 1) && (s == 1)

c1: x == 2
c2: k > 1
c3: s == 1

c1	c2	c3	
T	T	T	Test#11
T	T	F	Test#25
T	F	T	Test#26
T	F	F	Test#26
F	T	T	Test#15
F	T	F	Test#27
F	F	T	Test#28
F	F	F	Test#28

For Condition: (x == 2) && (k == 1) && (s == 1)

c1: x == 2
c2: k == 1
c3: s == 1

c1	c2	c3	
T	T	T	Test#1
T	T	F	Test#29
T	F	T	Test#26
T	F	F	Test#26
F	T	T	Test#9
F	T	F	Test#30
F	F	T	Test#28
F	F	F	Test#28

For Condition: (x == 2) && (k1 == 1) && (s == 2)

c1: x == 2
c2: k1 == 1
c3: s == 2

c1	c2	c3	
T	T	T	Test#4
T	T	F	Test#29
T	F	T	Test#26
T	F	F	Test#26
F	T	T	Test#10
F	T	F	Test#30
F	F	T	Test#28
F	F	F	Test#28

For Condition: (x == 3) && (k1 == 1) && (s == 2)

c1: x == 3
c2: k1 == 1
c3: s == 2

c1	c2	c3		Reason for non-executable
T	T	T	Test#10	
T	T	F	Test#30	
T	F	T	Test#28	
T	F	F	Test#28	
F	T	T	Non- executable	Here, x!=3. That means that x==2 because at the beginning of this method, "(x == 2) || (x == 3)" is checked. But, when "(x==2 && k1==1 && s==2)", this condition is already ***checked before*** reaching this point of code. We can reach "else if" in the code only if the conditions above this condition are false. Hence, this makes (F, T, T) non-executable
F	T	F	Test#29	
F	F	T	Test#26	
F	F	F	Test#26	

For Condition: (x == 3) && (k == 1) && (s == 1)

c1: x == 3
c2: k == 1
c3: s == 1

c1	c2	c3		Reason for non-excutable
T	T	T	Test#9	
T	T	F	Test#30	
T	F	T	Test#28	
T	F	F	Test#28	
F	T	T	Non- executable	Here, x!=3. That means that x==2 because at the beginning of this method, "(x == 2) || (x == 3)" is checked. But, when "(x==2 && k==1 && s==1)", this condition is already ***checked before*** reaching this point of code. We can reach "else if" in the code only if the conditions above this condition are false. Hence, this makes (F, T, T) non-executable
F	T	F	Test#29	
F	F	T	Test#26	
F	F	F	Test#26	

For Condition: (x == 3) && (k1 > 1) && (s == 2)

c1: x == 3
c2: k1 > 1
c3: s == 2

c1	c2	c3	
T	T	T	Test#10
T	T	F	Test#27
T	F	T	Test#28
T	F	F	Test#28
F	T	T	Test#3
F	T	F	Test#25
F	F	T	Test#26
F	F	F	Test#26

For Condition: (x == 3) && (k > 1) && (s == 1)

c1: x == 3
c2: k > 1
c3: s == 1

c1	c2	c3		Reason for non-executable
T	T	T	Test#15	
T	T	F	Test#27	
T	F	T	Test#28	
T	F	F	Test#28	
F	T	T	Non- executable	Here, x!=3. That means that x==2 because at the beginning of this method, "(x == 2) || (x == 3)" is checked. But, when "(x==2 && k>1 && s==1)", this condition is already ***checked before*** reaching this point of code. We can reach "else if" in the code only if the conditions above this condition are false. Hence, this makes (F, T, T) non-executable
F	T	F	Test#25	
F	F	T	Test#26	
F	F	F	Test#26	

Method: insert_large_cups()

For Condition: (x == 1) && (n > 0)

c1: x == 1
c2: n > 0

c1	c2	
T	T	Test#1
T	F	Test#23
F	T	Test#1
F	F	Test#23

For Condition: (x == 5) && (n > 0)

c1: x == 5

c2: n > 0

c1	**c2**	
T	T	Test#1
T	F	Test#31
F	T	Test#31
F	F	Test#23

Method: insert_small_cups()

For Condition: (x == 1) && (n > 0)

c1: x == 1

c2: n > 0

c1	**c2**	
T	T	Test#3
T	F	Test#23
F	T	Test#4
F	F	Test#23

For Condition: (x == 4) && (n > 0)

c1: x == 4

c2: n > 0

c1	**c2**	
T	T	Test#4
T	F	Test#31
F	T	Test#31
F	F	Test#23

Method: set_price()

For Condition: (x == 1) && (p > 0)

c1: x == 1

c2: p > 0

c1	**c2**	
T	T	Test#1
T	F	Test#23
F	T	Test#23
F	F	Test#23

Method: cancel()

For Condition: (x == 2) || (x == 3)
c1: x == 1
c2: n > 0

c1	c2		Reason for non-executable
T	T	Non- executable	value of “x” cannot be “2” and “3” simultaneously
T	F	Test#3	
F	T	Test#15	
F	F	Test#23	

Method: dispose()

For Condition: (x == 1)

T	Test#23
F	Test#23

4. Test Suit and the Results of its Execution

Test Suit:

Test#1: set_price 100 insert_large_cups 1 coin coin coin coin large_cup tea insert_large_cups 1 dispose
Test#2: insert_large_cups 2 set_price 125 coin dispose
Test#3: set_price 75 insert_small_cups 2 coin coin coin coin small_cup small_cup tea coin coin coin small_cup cancel dispose
Test#4: insert_small_cups 1 set_price 50 insert_large_cups 2 coin coin large_cup tea coin coin small_cup tea coin insert_small_cups 1 dispose
Test#5: dispose coin
Test#6: insert_large_cups 2 insert_large_cups 1 insert_small_cups 2 insert_small_cups 1 set_price 25 coin cancel coin coin cancel coin sugar coin small_cup sugar tea dispose
Test#7: set_price 25 insert_small_cups 1 insert_large_cups 2 dispose
Test#8: set_price 25 insert_small_cups 1 dispose
Test#9: set_price 50 set_price 25 insert_large_cups 1 coin coin sugar sugar sugar large_cup tea coin coin insert_large_cups 2 set_price 50 dispose
Test#10: set_price 25 insert_small_cups 1 coin small_cup sugar small_cup tea coin coin insert_small_cups 2 coin sugar small_cup tea dispose
Test#11: set_price 100 coin insert_large_cups 2 coin insert_small_cups 1 coin set_price 125 coin coin coin coin large_cup large_cup coin tea set_price 25 coin coin small_cup coin tea insert_small_cups 1 set_price 50 coin coin large_cup coin tea coin insert_large_cups 2 insert_small_cups 1 dispose
Test#12: set_price 25 insert_small_cups 1 coin sugar coin coin small_cup tea insert_small_cups 2 insert_large_cups 1 coin sugar large_cup coin large_cup tea insert_large_cups 1 insert_large_cups 1 dispose
Test#13: set_price 25 insert_small_cups 3 coin large_cup cancel coin large_cup small_cup tea coin large_cup sugar sugar coin sugar sugar small_cup large_cup small_cup coin tea dispose
Test#14 set_price 25 insert_large_cups 2 coin sugar sugar cancel coin sugar sugar large_cup sugar sugar tea dispose
Test#15: set_price 50 insert_large_cups 2 coin coin sugar cancel coin coin sugar large_cup coin tea insert_small_cups 2 coin coin sugar large_cup coin tea insert_large_cups 2 coin coin sugar small_cup small_cup coin tea insert_small_cups 1 dispose
Test#16: set_price 25 insert_large_cups 1 coin large_cup sugar sugar tea insert_large_cups 2 coin sugar large_cup cancel insert_small_cups 1 coin small_cup sugar sugar tea insert_small_cups 1 insert_small_cups 1 dispose
Test#17: set_price 50 insert_small_cups 2 coin coin small_cup sugar tea insert_large_cups 2 coin coin large_cup sugar tea coin coin small_cup sugar tea insert_small_cups 1 coin coin large_cup sugar tea insert_large_cups 1 dispose
Test#18: set_price 25 insert_small_cups 1 insert_large_cups 2 coin sugar small_cup large_cup small_cup coin cancel coin sugar small_cup coin tea insert_small_cups 1 coin sugar large_cup large_cup sugar sugar large_cup tea insert_large_cups 1 dispose
Test#19: set_price 25 insert_large_cups 2 coin sugar coin sugar sugar small_cup cancel set_price 25 coin cancel set_price 50 coin coin cancel coin coin cancel insert_small_cups 1 coin coin cancel insert_large_cups 1 coin coin cancel dispose
Test#20: set_price 25 insert_small_cups 10 coin small_cup tea insert_large_cups 1 coin small_cup tea insert_small_cups 1 coin small_cup tea set_price 25 coin small_cup tea coin sugar small_cup tea coin sugar small_cup tea set_price 50 coin coin sugar small_cup tea coin coin cancel dispose
Test#21: set_price 25 insert_large_cups 10 coin large_cup tea insert_large_cups 1 coin large_cup tea insert_small_cups 1 coin large_cup tea coin sugar large_cup tea coin sugar large_cup tea set_price 50 coin coin large_cup tea dispose
Test#22: set_price 100 insert_small_cups 1 coin coin coin coin sugar cancel insert_large_cups 1 coin coin coin coin sugar cancel dispose
Test#23: cancel insert_small_cups -1 insert_large_cups -3 set_price -10 set_price 50 coin coin insert_small_cups -1 insert_large_cups -2 dispose set_price 25 set_price -25 cancel dispose
Test#24: coin small_cup large_cup sugar tea dispose
Test#25: set_price 25 insert_small_cups 2 insert_large_cups 2 coin tea small_cup tea dispose
Test#26: set_price 50 coin coin tea small_cup tea large_cup tea cancel dispose
Test#27: set_price 100 insert_small_cups 2 insert_large_cups 2 coin coin coin coin sugar tea small_cup tea dispose
Test#28: set_price 25 coin sugar tea small_cup tea large_cup tea cancel dispose
Test#29: set_price 25 insert_small_cups 1 insert_large_cups 1 coin tea cancel dispose
Test#30: set_price 100 insert_small_cups 1 insert_large_cups 1 coin coin coin coin sugar tea cancel dispose
Test#31: set_price 25 insert_small_cups 1 insert_large_cups 1 coin insert_small_cups 1 small_cup tea insert_small_cups -1 insert_small_cups 1 coin insert_large_cups 2 large_cup tea insert_large_cups -2 insert_large_cups 2 dispose
Test#32: coin set_price -10 insert_large_cups 0 insert_small_cups -2 small_cup large_cup tea sugar cancel dispose

Test#33: set_price 25 insert_large_cups 1 coin large_cup tea set_price 50 small_cup large_cup tea sugar cancel dispose insert_small_cups 1 insert_large_cups 0 insert_large_cups 2 dispose
Test#34: set_price 25 insert_small_cups 1 coin small_cup tea set_price 50 small_cup large_cup tea sugar cancel dispose insert_large_cups 1 insert_small_cups 0 insert_small_cups 2 dispose
Test#35: set_price 25 coin set_price 50 dispose insert_large_cups 2 insert_small_cups 2 tea sugar set_price 50 dispose insert_large_cups 2 insert_small_cups 2 tea cancel dispose
$$ $$

Results of Test Suite Execution:

Test#1														**Result**
	Expected							**Actual**						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(100)	idle	1	100	0	0	0	0	idle	1	100	0	0	0	0	Pass
insert_large_cups(1)	idle	1	100	1	0	0	0	idle	1	100	1	0	0	0	Pass
coin()	idle	1	100	1	0	25	0	idle	1	100	1	0	25	0	Pass
coin()	idle	1	100	1	0	50	0	idle	1	100	1	0	50	0	Pass
coin()	idle	1	100	1	0	75	0	idle	1	100	1	0	75	0	Pass
coin()	coins inserted	2	100	1	0	0	0	coins inserted	2	100	1	0	0	0	Pass
large_cup	coins inserted	2	100	1	0	0	1	coins inserted	2	100	1	0	0	1	Pass
tea()	no large_cups	5	100	0	0	0	1	no large_cups	5	100	0	0	0	1	Pass
insert_large_cups(1)	idle	1	100	1	0	0	1	idle	1	100	1	0	0	1	Pass
dispose()	exit	6	100	1	0	0	1	exit	6	100	1	0	0	1	Pass

Test#2														**Result**
	Expected							**Actual**						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
insert_large_cups(2)	idle	1	0	2	0	0	0	idle	1	0	2	0	0	0	Pass
set_price(125)	idle	1	125	2	0	0	0	idle	1	125	2	0	0	0	Pass
coin()	idle	1	125	2	0	25	0	idle	1	125	2	0	25	0	Pass
dispose()	exit	6	125	2	0	25	0	exit	6	125	2	0	25	0	Pass

Test#3														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(75)	idle	1	75	0	0	0	0	idle	1	75	0	0	0	0	Pass
insert_small_cups(2)	idle	1	75	0	2	0	0	idle	1	75	0	2	0	0	Pass
coin()	idle	1	75	0	2	25	0	idle	0	75	0	2	25	0	Pass
coin()	idle	1	75	0	2	50	0	idle	0	75	0	2	50	0	Pass
coin()	coins inserted	2	75	0	2	0	0	coins inserted	0	75	0	2	0	0	Pass
small_cup()	coins inserted	2	75	0	2	0	2	coins inserted	2	75	0	2	0	2	Pass
small_cup()	coins inserted	2	75	0	2	0	2	coins inserted	2	75	0	2	0	2	Pass
tea()	idle	1	75	0	1	0	2	idle	1	75	0	1	0	2	Pass
coin()	idle	1	75	0	1	25	2	idle	1	75	0	1	25	2	Pass
coin()	idle	1	75	0	1	50	2	idle	1	75	0	1	50	2	Pass
coin()	coins inserted	2	75	0	1	0	0	coins inserted	2	75	0	1	0	0	Pass
small_cup()	coins inserted	2	75	0	1	0	2	coins inserted	2	75	0	1	0	2	Pass
cancel()	idle	1	75	0	1	0	2	idle	1	75	0	1	0	2	Pass
dispose()	exit	6	75	0	1	0	2	exit	6	75	0	1	0	2	Pass

Test#4														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
insert_small_cups(1)	idle	1	0	0	1	0	0	idle	1	0	0	1	0	0	Pass
set_price(50)	idle	1	50	0	1	0	0	idle	1	50	0	1	0	0	Pass
insert_large_cups(2)	idle	1	50	2	1	0	0	idle	1	50	2	1	0	0	Pass
coin()	idle	1	50	2	1	25	0	idle	1	50	2	1	25	0	Pass
coin()	coins inserted	2	50	2	1	0	0	coins inserted	2	50	2	1	0	0	Pass
large_cup()	coins inserted	2	50	2	1	0	1	coins inserted	2	50	2	1	0	1	Pass
tea()	idle	1	50	1	1	0	1	idle	1	50	1	1	0	1	Pass
coin()	idle	1	50	1	1	25	1	idle	1	50	1	1	25	1	Pass
coin()	coins inserted	2	50	1	1	0	1	coins inserted	2	50	1	1	0	1	Pass
small_cup()	coins inserted	2	50	1	1	0	2	coins inserted	2	50	1	1	0	2	Pass
tea()	no small cups	4	50	1	0	0	2	no small cups	4	50	1	0	0	2	Pass
coin()	no small cups	4	50	1	0	0	2	no small cups	4	50	1	0	0	2	Pass
insert_small_cups(1)	idle	1	50	1	1	0	2	idle	1	50	1	1	0	2	Pass
dispose()	exit	6	50	1	1	0	2	exit	6	50	1	1	0	2	Pass

Test#5															Result
	Expected							Actual							
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
dispose()	exit	6	0	0	0	0	0	exit	6	0	0	0	0	0	Pass
coin()	exit	6	0	0	0	0	0	exit	0	0	0	0	0	0	Pass

Test#6															Result
	Expected							Actual							
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
insert_large_cups(2)	idle	1	0	2	0	0	0	idle	1	0	2	0	0	0	Pass
insert_large_cups(1)	idle	1	0	3	0	0	0	idle	1	0	3	0	0	0	Pass
insert_small_cups(2)	idle	1	0	3	2	0	0	idle	1	0	3	2	0	0	Pass
insert_small_cups(1)	idle	1	0	3	3	0	0	idle	1	0	3	3	0	0	Pass
set_price(25)	idle	1	25	3	3	0	0	idle	1	25	3	3	0	0	Pass
coin()	coins inserted	2	25	3	3	0	0	coins inserted	2	25	3	3	0	0	Pass
cancel()	idle	1	25	3	3	0	0	idle	1	25	3	3	0	0	Pass
coin()	coins inserted	2	25	3	3	0	0	coins inserted	2	25	3	3	0	0	Pass
coin()	coins inserted	2	25	3	3	0	0	coins inserted	2	25	3	3	0	0	Pass
cancel()	idle	1	25	3	3	0	0	idle	1	25	3	3	0	0	Pass
coin()	coins inserted	2	25	3	3	0	0	coins inserted	2	25	3	3	0	0	Pass
sugar()	sugar	3	25	3	3	0	0	sugar	3	25	3	3	0	0	Pass
coin()	sugar	3	25	3	3	0	0	sugar	3	25	3	3	0	0	Pass
small_cup()	sugar	3	25	3	3	0	2	sugar	3	25	3	3	0	2	Pass
sugar()	coins inserted	2	25	3	3	0	2	coins inserted	2	25	3	3	0	2	Pass
tea()	idle	1	25	3	2	0	2	idle	1	25	3	2	0	2	Pass
dispose()	exit	6	25	3	2	0	2	exit	6	25	3	2	0	2	Pass

Test#7															Result
	Expected							Actual							
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(1)	idle	1	25	0	1	0	0	idle	1	25	0	1	0	0	Pass
insert_large_cups(2)	idle	1	25	2	1	0	0	idle	1	25	2	1	0	0	Pass
dispose()	exit	6	25	2	1	0	0	exit	6	25	2	1	0	0	Pass

Test#8															Result
	Expected							Actual							
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(1)	idle	1	25	0	1	0	0	idle	1	25	0	1	0	0	Pass
dispose()	exit	6	25	0	1	0	0	exit	6	25	0	1	0	0	Pass

Test#9														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(50)	idle	1	50	0	0	0	0	idle	1	50	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_large_cups(1)	idle	1	25	1	0	0	0	idle	1	25	1	0	0	0	Pass
coin()	coins inserted	2	25	1	0	0	0	coins inserted	2	25	1	0	0	0	Pass
coin()	coins inserted	2	25	1	0	0	0	coins inserted	2	25	1	0	0	0	Pass
sugar()	sugar	3	25	1	0	0	0	sugar	3	25	1	0	0	0	Pass
sugar()	coins inserted	2	25	1	0	0	0	coins inserted	2	25	1	0	0	0	Pass
sugar()	sugar	3	25	1	0	0	0	sugar	3	25	1	0	0	0	Pass
large_cup()	sugar	3	25	1	0	0	1	sugar	3	25	1	0	0	1	Pass
tea()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
coin()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
coin()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
insert_large_cups(2)	idle	1	25	2	0	0	1	idle	1	25	2	0	0	1	Pass
set_price(50)	idle	1	50	2	0	0	1	idle	1	50	2	0	0	1	Pass
dispose()	exit	6	50	2	0	0	1	exit	6	50	2	0	0	1	Pass

Test#10														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(1)	idle	1	25	0	1	0	0	idle	1	25	0	1	0	0	Pass
coin()	coins inserted	2	25	0	1	0	0	coins inserted	2	25	0	1	0	0	Pass
small_cup()	coins inserted	2	25	0	1	0	2	coins inserted	2	25	0	1	0	2	Pass
sugar()	sugar	3	25	0	1	0	2	sugar	3	25	0	1	0	2	Pass
small_cup()	sugar	3	25	0	1	0	2	sugar	3	25	0	1	0	2	Pass
tea()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
coin()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
coin()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
insert_small_cups(2)	idle	1	25	0	2	0	2	idle	1	25	0	2	0	2	Pass
coin()	coins inserted	2	25	0	2	0	0	coins inserted	2	25	0	2	0	0	Pass
sugar()	sugar	3	25	0	2	0	0	sugar	3	25	0	2	0	0	Pass
small_cup()	sugar	3	25	0	2	0	2	sugar	3	25	0	2	0	2	Pass
tea()	idle	1	25	0	1	0	2	idle	1	25	0	1	0	2	Pass
dispose()	exit	6	25	0	1	0	2	exit	6	25	0	1	0	2	Pass

Test#11															Result
	Expected							Actual							
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(100)	idle	1	100	0	0	0	0	idle	1	100	0	0	0	0	Pass
coin()	idle	1	100	2	0	25	0	idle	1	100	2	0	25	0	Pass
insert_large_cups(2)	idle	1	100	2	0	25	0	idle	1	100	2	0	25	0	Pass
coin()	idle	1	100	2	0	50	0	idle	1	100	2	0	50	0	Pass
insert_small_cups(1)	idle	1	100	2	1	50	0	idle	1	100	2	1	50	0	Pass
coin()	idle	1	100	2	1	75	0	idle	1	100	2	1	75	0	Pass
set_price(125)	idle	1	125	2	1	75	0	idle	1	125	2	1	75	0	Pass
coin()	idle	1	125	2	1	100	0	idle	1	125	2	1	100	0	Pass
coin()	idle	1	125	2	1	0	0	idle	1	125	2	1	0	0	Pass
coin()	coins inserted	2	125	2	1	0	0	coins inserted	2	125	2	1	0	0	Pass
coin()	coins inserted	2	125	2	1	0	0	coins inserted	2	125	2	1	0	0	Pass
large_cup()	coins inserted	2	125	2	1	0	1	coins inserted	2	125	2	1	0	1	Pass
large_cup()	coins inserted	2	125	2	1	0	1	coins inserted	2	125	2	1	0	1	Pass
coin()	coins inserted	2	125	2	1	0	1	coins inserted	2	125	2	1	0	1	Pass
tea()	idle	1	125	1	1	0	1	idle	1	125	1	1	0	1	Pass
set_price(25)	idle	1	25	1	1	0	1	idle	1	25	1	1	0	1	Pass
coin()	coins inserted	2	25	1	1	0	0	coins inserted	2	25	1	1	0	0	Pass
coin()	coins inserted	2	25	1	1	0	0	coins inserted	2	25	1	1	0	0	Pass
small_cup()	coins inserted	2	25	1	1	0	2	coins inserted	2	25	1	1	0	2	Pass
coin()	coins inserted	2	25	1	1	0	2	coins inserted	2	25	1	1	0	2	Pass
tea()	no small cups	4	25	1	0	0	2	no small cups	4	25	1	0	0	2	Pass
insert_small_cups(1)	idle	1	25	1	1	0	2	idle	1	25	1	1	0	2	Pass
set_price(50)	idle	1	50	1	1	0	2	idle	1	50	1	1	0	2	Pass
coin()	idle	1	50	1	1	25	2	idle	1	50	1	1	25	2	Pass
coin()	coins inserted	2	50	1	1	0	0	coins inserted	2	50	1	1	0	0	Pass
large_cup()	coins inserted	2	50	1	1	0	1	coins inserted	2	50	1	1	0	1	Pass
coin()	coins inserted	2	50	1	1	0	1	coins inserted	2	50	1	1	0	1	Pass
tea()	no large_cups	5	50	0	1	0	1	no large_cups	5	50	0	1	0	1	Pass
coin()	no large_cups	5	50	0	1	0	1	no large_cups	5	50	0	1	0	1	Pass
insert_large_cups(2)	idle	1	50	2	1	0	1	idle	1	50	2	1	0	1	Pass
insert_small_cups(1)	idle	1	50	2	2	0	1	idle	1	50	2	2	0	1	Pass
dispose()	exit	6	50	2	2	0	1	exit	6	50	2	2	0	1	Pass

Test#12															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(1)	idle	1	25	0	1	0	0	idle	1	25	0	1	0	0	Pass
coin()	coins inserted	2	25	0	1	0	0	coins inserted	2	25	0	1	0	0	Pass
sugar()	sugar	3	25	0	1	0	0	sugar	3	25	0	1	0	0	Pass
coin()	sugar	3	25	0	1	0	0	sugar	3	25	0	1	0	0	Pass
coin()	sugar	3	25	0	1	0	0	sugar	3	25	0	1	0	0	Pass
small_cup()	sugar	3	25	0	1	0	2	sugar	3	25	0	1	0	2	Pass
tea()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
insert_small_cups(2)	idle	1	25	0	2	0	2	idle	1	25	0	2	0	2	Pass
insert_large_cups(1)	idle	1	25	1	2	0	2	idle	1	25	1	2	0	2	Pass
coin()	coins inserted	2	25	1	2	0	0	coins inserted	2	25	1	2	0	0	Pass
sugar()	sugar	3	25	1	2	0	0	sugar	3	25	1	2	0	0	Pass
large_cup()	sugar	3	25	1	2	0	1	sugar	3	25	1	2	0	1	Pass
coin()	sugar	3	25	1	2	0	1	sugar	3	25	1	2	0	1	Pass
large_cup()	sugar	3	25	1	2	0	1	sugar	3	25	1	2	0	1	Pass
tea()	no large_cups	5	25	0	2	0	1	no large_cups	5	25	0	2	0	1	Pass
insert_large_cups(1)	idle	1	25	1	2	0	1	idle	1	25	1	2	0	1	Pass
insert_large_cups(1)	idle	1	25	2	2	0	1	idle	1	25	2	2	0	1	Pass
dispose()	exit	6	25	2	2	0	1	exit	6	25	2	2	0	1	Pass

Test#13															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(3)	idle	1	25	0	3	0	0	idle	1	25	0	3	0	0	Pass
coin()	coins inserted	2	25	0	3	0	0	coins inserted	2	25	0	3	0	0	Pass
large_cup()	coins inserted	2	25	0	3	0	1	coins inserted	2	25	0	3	0	1	Pass
cancel()	idle	1	25	0	3	0	1	idle	1	25	0	3	0	1	Pass
coin()	coins inserted	2	25	0	3	0	0	coins inserted	2	25	0	3	0	0	Pass
large_cup()	coins inserted	2	25	0	3	0	1	coins inserted	2	25	0	3	0	1	Pass
small_cup()	coins inserted	2	25	0	3	0	2	coins inserted	2	25	0	3	0	2	Pass
tea()	idle	1	25	0	2	0	2	idle	1	25	0	2	0	2	Pass
coin()	coins inserted	2	25	0	2	0	0	coins inserted	2	25	0	2	0	0	Pass
large_cup()	coins inserted	2	25	0	2	0	1	coins inserted	2	25	0	2	0	1	Pass
sugar()	sugar	3	25	0	2	0	1	sugar	3	25	0	2	0	1	Pass
sugar()	coins inserted	2	25	0	2	0	1	coins inserted	2	25	0	2	0	1	Pass
coin()	coins inserted	2	25	0	2	0	1	coins inserted	2	25	0	2	0	1	Pass
sugar()	sugar	3	25	0	2	0	1	sugar	3	25	0	2	0	1	Pass
sugar()	coins inserted	2	25	0	2	0	1	coins inserted	2	25	0	2	0	1	Pass
small_cup()	coins inserted	2	25	0	2	0	2	coins inserted	2	25	0	2	0	2	Pass
large_cup()	coins inserted	2	25	0	2	0	1	coins inserted	2	25	0	2	0	1	Pass
small_cup()	coins inserted	2	25	0	2	0	2	coins inserted	2	25	0	2	0	2	Pass
coin()	coins inserted()	2	25	0	2	0	2	coins inserted()	2	25	0	2	0	2	Pass
tea()	idle	1	25	0	1	0	2	idle	1	25	0	1	0	2	Pass
dispose()	exit()	6	25	0	1	0	2	exit()	6	25	0	1	0	2	Pass

Test#14															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_large_cups(2)	idle	1	25	2	0	0	0	idle	1	25	2	0	0	0	Pass
coin()	coins inserted	2	25	2	0	0	0	coins inserted	2	25	2	0	0	0	Pass
sugar()	sugar	3	25	2	0	0	0	sugar	3	25	2	0	0	0	Pass
sugar()	coins inserted	2	25	2	0	0	0	coins inserted	2	25	2	0	0	0	Pass
cancel()	idle	1	25	2	0	0	0	idle	1	25	2	0	0	0	Pass
coin()	coins inserted	2	25	2	0	0	0	coins inserted	2	25	2	0	0	0	Pass
sugar()	sugar	3	25	2	0	0	0	sugar	3	25	2	0	0	0	Pass
sugar()	coins inserted	2	25	2	0	0	0	coins inserted	2	25	2	0	0	0	Pass
large_cup()	coins inserted	2	25	2	0	0	1	coins inserted	2	25	2	0	0	1	Pass
sugar()	sugar	3	25	2	0	0	1	sugar	3	25	2	0	0	1	Pass
sugar()	coins inserted	2	25	2	0	0	1	coins inserted	2	25	2	0	0	1	Pass
tea()	idle	1	25	1	0	0	1	idle	1	25	1	0	0	1	Pass
dispose()	exit	6	25	1	0	0	1	exit	6	25	1	0	0	1	Pass

Test#15															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(50)	idle	1	50	0	0	0	0	idle	1	50	0	0	0	0	Pass
insert_large_cups(2)	idle	1	50	2	0	0	0	idle	1	50	2	0	0	0	Pass
coin()	idle	1	50	2	0	25	0	idle	1	50	2	0	25	0	Pass
coin()	coins inserted	2	50	2	0	0	0	coins inserted	2	50	2	0	0	0	Pass
sugar()	sugar	3	50	2	0	0	0	sugar	3	50	2	0	0	0	Pass
cancel()	idle	1	50	2	0	0	0	idle	1	50	2	0	0	0	Pass
coin()	idle	1	50	2	0	25	0	idle	1	50	2	0	25	0	Pass
coin()	coins inserted	2	50	2	0	0	0	coins inserted	2	50	2	0	0	0	Pass
sugar()	sugar	3	50	2	0	0	0	sugar	3	50	2	0	0	0	Pass
large_cup()	sugar	3	50	2	0	0	1	sugar	3	50	2	0	0	1	Pass
coin()	sugar	3	50	2	0	0	1	sugar	3	50	2	0	0	1	Pass
tea()	idle	1	50	1	0	0	1	idle	1	50	1	0	0	1	Pass
insert_small_cups(2)	idle	1	50	1	2	0	1	idle	1	50	1	2	0	1	Pass
coin()	idle	1	50	1	2	25	1	idle	1	50	1	2	25	1	Pass
coin()	coins inserted	2	50	1	2	0	1	coins inserted	2	50	1	2	0	1	Pass
sugar()	sugar	3	50	1	2	0	1	sugar	3	50	1	2	0	1	Pass
large_cup()	sugar	3	50	1	2	0	1	sugar	3	50	1	2	0	1	Pass
coin()	sugar	3	50	1	2	0	1	sugar	3	50	1	2	0	1	Pass
tea()	no large_cups	5	50	0	2	0	1	no large_cups	5	50	0	2	0	1	Pass
insert_large_cups(2)	idle	1	50	2	2	0	1	idle	1	50	2	2	0	1	Pass
coin()	idle	1	50	2	2	25	1	idle	1	50	2	2	25	1	Pass
coin()	coins inserted	2	50	2	2	0	0	coins inserted	2	50	2	2	0	0	Pass
sugar()	sugar	3	50	2	2	0	0	sugar	3	50	2	2	0	0	Pass
small_cup()	sugar	3	50	2	2	0	2	sugar	3	50	2	2	0	2	Pass
small_cup()	sugar	3	50	2	2	0	2	sugar	3	50	2	2	0	2	Pass
coin()	sugar	3	50	2	2	0	2	sugar	3	50	2	2	0	2	Pass
tea()	idle	1	50	2	1	0	2	idle	1	50	2	1	0	2	Pass
insert_small_cups(1)	idle	1	50	2	2	0	2	idle	1	50	2	2	0	2	Pass
dispose	exit	6	50	2	2	0	2	exit	6	50	2	2	0	2	Pass

Test#16															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_large_cups(1)	idle	1	25	1	0	0	0	idle	1	25	1	0	0	0	Pass
coin()	coins inserted	2	25	1	0	0	0	coins inserted	2	25	1	0	0	0	Pass
large_cup()	coins inserted	2	25	1	0	0	1	coins inserted	2	25	1	0	0	1	Pass
sugar()	sugar	3	25	1	0	0	1	sugar	3	25	1	0	0	1	Pass
sugar()	coins inserted	2	25	1	0	0	1	coins inserted	2	25	1	0	0	1	Pass
tea()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
insert_large_cups(2)	idle	1	25	2	0	0	1	idle	1	25	2	0	0	1	Pass
coin()	coins inserted	2	25	2	0	0	0	coins inserted	2	25	2	0	0	0	Pass
sugar()	sugar	3	25	2	0	0	0	sugar	3	25	2	0	0	0	Pass
large_cup()	sugar	3	25	2	0	0	1	sugar	3	25	2	0	0	1	Pass
cancel()	idle	1	25	2	0	0	1	idle	1	25	2	0	0	1	Pass
insert_small_cups(1)	idle	1	25	2	1	0	1	idle	1	25	2	1	0	1	Pass
coin()	coins inserted	2	25	2	1	0	0	coins inserted	2	25	2	1	0	0	Pass
small_cup()	coins inserted	2	25	2	1	0	2	coins inserted	2	25	2	1	0	2	Pass
sugar()	sugar	3	25	2	1	0	2	sugar	3	25	2	1	0	2	Pass
sugar()	coins inserted	2	25	2	1	0	2	coins inserted	2	25	2	1	0	2	Pass
tea()	no small cups	4	25	2	0	0	2	no small cups	4	25	2	0	0	2	Pass
insert_small_cups(1)	idle	1	25	2	1	0	2	idle	1	25	2	1	0	2	Pass
insert_small_cups(1)	idle	1	25	2	2	0	2	idle	1	25	2	2	0	2	Pass
dispose()	exit	6	25	2	2	0	2	exit	6	25	2	2	0	2	Pass

Test#17															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(50)	idle	1	50	0	0	0	0	idle	1	50	0	0	0	0	Pass
insert_small_cups(2)	idle	1	50	0	2	0	0	idle	1	50	0	2	0	0	Pass
coin()	idle	1	50	0	2	25	0	idle	1	50	0	2	25	0	Pass
coin()	coins inserted	2	50	0	2	0	0	coins inserted	2	50	0	2	0	0	Pass
small_cup()	coins inserted	2	50	0	2	0	2	coins inserted	2	50	0	2	0	2	Pass
sugar()	sugar	3	50	0	2	0	2	sugar	3	50	0	2	0	2	Pass
tea()	idle	1	50	0	1	0	2	idle	1	50	0	1	0	2	Pass
insert_large_cups(2)	idle	1	50	2	1	0	2	idle	1	50	2	1	0	2	Pass
coin()	idle	1	50	2	1	25	2	idle	1	50	2	1	25	2	Pass
coin()	coins inserted	2	50	2	1	0	0	coins inserted	2	50	2	1	0	0	Pass
large_cup()	coins inserted	2	50	2	1	0	1	coins inserted	2	50	2	1	0	1	Pass
sugar()	sugar	3	50	2	1	0	1	sugar	3	50	2	1	0	1	Pass
tea()	idle	1	50	1	1	0	1	idle	1	50	1	1	0	1	Pass
coin()	idle	1	50	1	1	25	1	idle	1	50	1	1	25	1	Pass
coin()	coins inserted	2	50	1	1	0	0	coins inserted	2	50	1	1	0	0	Pass
small_cup()	coins inserted	2	50	1	1	0	2	coins inserted	2	50	1	1	0	2	Pass
sugar()	sugar	3	50	1	1	0	2	sugar	3	50	1	1	0	2	Pass
tea()	no small cups	4	50	1	0	0	2	no small cups	4	50	1	0	0	2	Pass
insert_small_cups(1)	idle	1	50	1	1	0	2	idle	1	50	1	1	0	2	Pass
coin()	idle	1	50	1	1	25	2	idle	1	50	1	1	25	2	Pass
coin()	coins inserted	2	50	1	1	0	0	coins inserted	2	50	1	1	0	0	Pass
large_cup()	coins inserted	2	50	1	1	0	1	coins inserted	2	50	1	1	0	1	Pass
sugar()	sugar	3	50	1	1	0	1	sugar	3	50	1	1	0	1	Pass
tea()	no large_cup	5	50	0	1	0	1	no large_cup	5	50	0	1	0	1	Pass
insert_large_cups(1)	idle	1	50	1	1	0	1	idle	1	50	1	1	0	1	Pass
dispose()	exit	6	50	1	1	0	1	exit	6	50	1	1	0	1	Pass

Test#18															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(1)	idle	1	25	0	1	0	0	idle	1	25	0	1	0	0	Pass
insert_large_cups(2)	idle	1	25	2	1	0	0	idle	1	25	2	1	0	0	Pass
coin()	coins inserted	2	25	2	1	0	0	coins inserted	2	25	2	1	0	0	Pass
sugar()	sugar	3	25	2	1	0	0	sugar	3	25	2	1	0	0	Pass
small_cup()	sugar	3	25	2	1	0	2	sugar	3	25	2	1	0	2	Pass
large_cup()	sugar	3	25	2	1	0	1	sugar	3	25	2	1	0	1	Pass
small_cup()	sugar	3	25	2	1	0	2	sugar	3	25	2	1	0	2	Pass
coin()	sugar	3	25	2	1	0	2	sugar	3	25	2	1	0	2	Pass
cancel()	idle	1	25	2	1	0	2	idle	1	25	2	1	0	2	Pass
coin()	coins inserted	2	25	2	1	0	0	coins inserted	2	25	2	1	0	0	Pass
sugar()	sugar	3	25	2	1	0	0	sugar	3	25	2	1	0	0	Pass
small_cup()	sugar	3	25	2	1	0	2	sugar	3	25	2	1	0	2	Pass
coin()	sugar	3	25	2	1	0	2	sugar	3	25	2	1	0	2	Pass
tea()	no small cups	4	25	2	0	0	2	no small cups	4	25	2	0	0	2	Pass
insert_small_cups(1)	idle	1	25	2	1	0	2	idle	1	25	2	1	0	2	Pass
coin()	coins inserted	2	25	2	1	0	0	coins inserted	2	25	2	1	0	0	Pass
sugar()	sugar	3	25	2	1	0	0	sugar	3	25	2	1	0	0	Pass
large_cup()	sugar	3	25	2	1	0	1	sugar	3	25	2	1	0	1	Pass
large_cup()	sugar	3	25	2	1	0	1	sugar	3	25	2	1	0	1	Pass
sugar()	coins inserted	2	25	2	1	0	1	coins inserted	2	25	2	1	0	1	Pass
sugar()	sugar	3	25	2	1	0	1	sugar	3	25	2	1	0	1	Pass
large_cup()	sugar	3	25	2	1	0	1	sugar	3	25	2	1	0	1	Pass
tea()	idle	1	25	1	1	0	1	idle	1	25	1	1	0	1	Pass
insert_large_cups(1)	idle	1	25	2	1	0	1	idle	1	25	2	1	0	1	Pass
dispose()	exit	6	25	2	1	0	1	exit	6	25	2	1	0	1	Pass

Test#19															Result
	Expected							Actual							
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_large_cups(2)	idle	1	25	2	0	0	0	idle	1	25	2	0	0	0	Pass
coin()	coins inserted	2	25	2	0	0	0	coins inserted	2	25	2	0	0	0	Pass
sugar()	sugar	3	25	2	0	0	0	sugar	3	25	2	0	0	0	Pass
coin()	sugar	3	25	2	0	0	0	sugar	3	25	2	0	0	0	Pass
sugar()	coins inserted	2	25	2	0	0	0	coins inserted	2	25	2	0	0	0	Pass
sugar()	sugar	3	25	2	0	0	0	sugar	3	25	2	0	0	0	Pass
small_cup()	sugar	3	25	2	0	0	2	sugar	3	25	2	0	0	2	Pass
cancel()	idle	1	25	2	0	0	2	idle	1	25	2	0	0	2	Pass
set_price(25)	idle	1	25	2	0	0	2	idle	1	25	2	0	0	2	Pass
coin()	coins inserted	2	25	2	0	0	0	coins inserted	2	25	2	0	0	0	Pass
cancel()	idle	1	25	2	0	0	0	idle	1	25	2	0	0	0	Pass
set_price(50)	idle	1	50	2	0	0	0	idle	1	50	2	0	0	0	Pass
coin()	idle	1	50	2	0	25	0	idle	1	50	2	0	25	0	Pass
coin()	coins inserted	2	50	2	0	0	0	coins inserted	2	50	2	0	0	0	Pass
cancel()	idle	1	50	2	0	0	0	idle	1	50	2	0	0	0	Pass
coin()	idle	1	50	2	0	25	0	idle	1	50	2	0	25	0	Pass
coin()	coins inserted	2	50	2	0	0	0	coins inserted	2	50	2	0	0	0	Pass
cancel()	idle	1	50	2	0	0	0	idle	1	50	2	0	0	0	Pass
insert_small_cups(1)	idle	1	50	2	1	0	0	idle	1	50	2	1	0	0	Pass
coin()	idle	1	50	2	1	25	0	idle	1	50	2	1	25	0	Pass
coin()	coins inserted	2	50	2	1	0	0	coins inserted	2	50	2	1	0	0	Pass
cancel()	idle	1	50	2	1	0	0	idle	1	50	2	1	0	0	Pass
insert_large_cups(1)	idle	1	50	3	1	0	0	idle	1	50	3	1	0	0	Pass
coin()	idle	1	50	3	1	25	0	idle	1	50	3	1	25	0	Pass
coin()	coins inserted	2	50	3	1	0	0	coins inserted	2	50	3	1	0	0	Pass
cancel()	idle	1	50	3	1	0	0	idle	1	50	3	1	0	0	Pass
dispose()	exit	6	50	3	1	0	0	exit	6	50	3	1	0	0	Pass

Test#20															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	1	25	0	0	0	0		Pass
insert_small_cups(10)	idle	1	25	0	10	0	0	1	25	0	10	0	0		Pass
coin()	coins inserted	2	25	0	10	0	0	2	25	0	10	0	0		Pass
small_cup()	coins inserted	2	25	0	10	0	2	2	25	0	10	0	2		Pass
tea()	idle	1	25	0	9	0	2	1	25	0	9	0	2		Pass
insert_large_cups(1)	idle	1	25	1	9	0	2	1	25	1	9	0	2		Pass
coin()	coins inserted	2	25	1	9	0	0	2	25	1	9	0	0		Pass
small_cup()	coins inserted	2	25	1	9	0	2	2	25	1	9	0	2		Pass
tea()	idle	1	25	1	8	0	2	1	25	1	8	0	2		Pass
insert_small_cups(1)	idle	1	25	1	9	0	2	1	25	1	9	0	2		Pass
coin()	coins inserted	2	25	1	9	0	0	2	25	1	9	0	0		Pass
small_cup()	coins inserted	2	25	1	9	0	2	2	25	1	9	0	2		Pass
tea()	idle	1	25	1	8	0	2	1	25	1	8	0	2		Pass
set_price(25)	idle	1	25	1	8	0	2	1	25	1	8	0	2		Pass
coin()	coins inserted	2	25	1	8	0	0	2	25	1	8	0	0		Pass
small_cup()	coins inserted	2	25	1	8	0	2	2	25	1	8	0	2		Pass
tea()	idle	1	25	1	7	0	2	1	25	1	7	0	2		Pass
coin()	coins inserted	2	25	1	7	0	0	2	25	1	7	0	0		Pass
sugar()	sugar	3	25	1	7	0	0	3	25	1	7	0	0		Pass
small_cup()	sugar	3	25	1	7	0	2	3	25	1	7	0	2		Pass
tea()	idle	1	25	1	6	0	2	1	25	1	6	0	2		Pass
coin()	coins inserted	2	25	1	6	0	0	2	25	1	6	0	0		Pass
sugar()	sugar	3	25	1	6	0	0	3	25	1	6	0	0		Pass
small_cup()	sugar	3	25	1	6	0	2	3	25	1	6	0	2		Pass
tea()	idle	1	25	1	5	0	2	1	25	1	5	0	2		Pass
set_price(50)	idle	1	50	1	5	0	2	1	50	1	5	0	2		Pass
coin()	idle	1	50	1	5	25	2	1	50	1	5	25	2		Pass
coin()	coins inserted	2	50	1	5	0	0	2	50	1	5	0	0		Pass
sugar()	sugar	3	50	1	5	0	0	3	50	1	5	0	0		Pass
small_cup()	sugar	3	50	1	5	0	2	3	50	1	5	0	2		Pass
tea()	idle	1	50	1	4	0	2	1	50	1	4	0	2		Pass
coin()	idle	1	50	1	4	25	2	1	50	1	4	25	2		Pass
coin()	coins inserted	2	50	1	4	0	0	2	50	1	4	0	0		Pass
cancel()	idle	1	50	1	4	0	0	1	50	1	4	0	0		Pass
dispose()	exit	6	50	1	4	0	0	6	50	1	4	0	0		Pass

Test#21															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_large_cups(10)	idle	1	25	10	0	0	0	idle	1	25	10	0	0	0	Pass
coin()	coins inserted	2	25	10	0	0	0	coins inserted	2	25	10	0	0	0	Pass
large_cup()	coins inserted	2	25	10	0	0	1	coins inserted	2	25	10	0	0	1	Pass
tea()	idle	1	25	9	0	0	1	idle	1	25	9	0	0	1	Pass
insert_large_cups(1)	idle	1	25	10	0	0	1	idle	1	25	10	0	0	1	Pass
coin()	coins inserted	2	25	10	0	0	0	coins inserted	2	25	10	0	0	0	Pass
large_cup()	coins inserted	2	25	10	0	0	1	coins inserted	2	25	10	0	0	1	Pass
tea()	idle	1	25	9	0	0	1	idle	1	25	9	0	0	1	Pass
insert_small_cups(1)	idle	1	25	9	1	0	1	idle	1	25	9	1	0	1	Pass
coin()	coins inserted	2	25	9	1	0	0	coins inserted	2	25	9	1	0	0	Pass
large_cup()	coins inserted	2	25	9	1	0	1	coins inserted	2	25	9	1	0	1	Pass
tea()	idle	1	25	8	1	0	1	idle	1	25	8	1	0	1	Pass
coin()	coins inserted	2	25	8	1	0	0	coins inserted	2	25	8	1	0	0	Pass
sugar()	sugar	3	25	8	1	0	0	sugar	3	25	8	1	0	0	Pass
large_cup()	sugar	3	25	8	1	0	1	sugar	3	25	8	1	0	1	Pass
tea()	idle	1	25	7	1	0	1	idle	1	25	7	1	0	1	Pass
coin()	coins inserted	2	25	7	1	0	0	coins inserted	2	25	7	1	0	0	Pass
sugar()	sugar	3	25	7	1	0	0	sugar	3	25	7	1	0	0	Pass
large_cup()	sugar	3	25	7	1	0	1	sugar	3	25	7	1	0	1	Pass
tea()	idle	1	25	6	1	0	1	idle	1	25	6	1	0	1	Pass
set_price(50)	idle	1	50	6	1	0	1	idle	1	50	6	1	0	1	Pass
coin()	idle	1	50	6	1	25	1	idle	1	50	6	1	25	1	Pass
coin()	coins inserted	2	50	6	1	0	0	coins inserted	2	50	6	1	0	0	Pass
large_cup()	coins inserted	2	50	6	1	0	1	coins inserted	2	50	6	1	0	1	Pass
tea()	idle	1	50	5	1	0	1	idle	1	50	5	1	0	1	Pass
dispose()	exit	6	50	5	1	0	1	exit	6	50	5	1	0	1	Pass

Test#22														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(100)	idle	1	100	0	0	0	0	idle	1	100	0	0	0	0	Pass
insert_small_cups(1)	idle	1	100	0	1	0	0	idle	1	100	0	1	0	0	Pass
coin()	idle	1	100	0	1	25	0	idle	1	100	0	1	25	0	Pass
coin()	idle	1	100	0	1	50	0	idle	1	100	0	1	50	0	Pass
coin()	idle	1	100	0	1	75	0	idle	1	100	0	1	75	0	Pass
coin()	coins inserted	2	100	0	1	0	0	coins inserted	2	100	0	1	0	0	Pass
sugar()	sugar	3	100	0	1	0	0	sugar	3	100	0	1	0	0	Pass
cancel()	idle	1	100	0	1	0	0	idle	1	100	0	1	0	0	Pass
insert_large_cups(1)	idle	1	100	1	1	0	0	idle	1	100	1	1	0	0	Pass
coin()	idle	1	100	1	1	25	0	idle	1	100	1	1	25	0	Pass
coin()	idle	1	100	1	1	50	0	idle	1	100	1	1	50	0	Pass
coin()	idle	1	100	1	1	75	0	idle	1	100	1	1	75	0	Pass
coin()	coins inserted	2	100	1	1	0	0	coins inserted	2	100	1	1	0	0	Pass
sugar()	sugar	3	100	1	1	0	0	sugar	3	100	1	1	0	0	Pass
cancel()	idle	1	100	1	1	0	0	idle	1	100	1	1	0	0	Pass
dispose()	exit	6	100	1	1	0	0	exit	6	100	1	1	0	0	Pass

Test#23														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
cancel()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
insert_small_cups(-1)	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
insert_large_cups(-3)	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
set_price(-10)	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
set_price(50)	idle	1	50	0	0	0	0	idle	1	50	0	0	0	0	Pass
coin()	idle	1	50	0	0	25	0	idle	1	50	0	0	25	0	Pass
coin()	coins inserted	2	50	0	0	0	0	coins inserted	2	50	0	0	0	0	Pass
insert_small_cups(-1)	coins inserted	2	50	0	0	0	0	coins inserted	2	50	0	0	0	0	Pass
insert_large_cups(-2)	coins inserted	2	50	0	0	0	0	coins inserted	2	50	0	0	0	0	Pass
dispose()	coins inserted	2	50	0	0	0	0	coins inserted	2	50	0	0	0	0	Pass
set_price(25)	coins inserted	2	50	0	0	0	0	coins inserted	2	50	0	0	0	0	Pass
set_price(-25)	coins inserted	2	50	0	0	0	0	coins inserted	2	50	0	0	0	0	Pass
cancel()	idle	1	50	0	0	0	0	idle	1	50	0	0	0	0	Pass
dispose()	exit	6	50	0	0	0	0	exit	6	50	0	0	0	0	Pass

Test#24														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
coin()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
small_cup()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
large_cup()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
sugar()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
tea()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
dispose()	exit	6	0	0	0	0	0	exit	6	0	0	0	0	0	Pass

Test#25														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(2)	idle	1	25	0	2	0	0	idle	1	25	0	2	0	0	Pass
insert_large_cups(2)	idle	1	25	2		0	0	idle	1	25	2		0	0	Pass
coin()	coins inserted	2	25	2	2	0	0	coins inserted	2	25	2	2	0	0	Pass
tea()	coins inserted	2	25	2	2	0	0	coins inserted	2	25	2	2	0	0	Pass
small_cup()	coins inserted	2	25	2	2	0	2	coins inserted	2	25	2	2	0	2	Pass
tea()	idle	1	25	2	1	0	2	idle	1	25	2	1	0	2	Pass
dispose()	exit	6	25	2	1	0	2	exit	6	25	2	1	0	2	Pass

Test#26														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(50)	idle	1	50	0	0	0	0	idle	1	50	0	0	0	0	Pass
coin()	idle	1	50	0	0	25	0	idle	1	50	0	0	25	0	Pass
coin()	coins inserted	2	50	0	0	0	0	coins inserted	2	50	0	0	0	0	Pass
tea()	coins inserted	2	50	0	0	0	0	coins inserted	2	50	0	0	0	0	Pass
small_cup()	coins inserted	2	50	0	0	0	2	coins inserted	2	50	0	0	0	2	Pass
tea()	coins inserted	2	50	0	0	0	2	coins inserted	2	50	0	0	0	2	Pass
large_cup()	coins inserted	2	50	0	0	0	1	coins inserted	2	50	0	0	0	1	Pass
tea()	coins inserted	2	50	0	0	0	1	coins inserted	2	50	0	0	0	1	Pass
cancel()	idle	1	50	0	0	0	1	idle	1	50	0	0	0	1	Pass
dispose()	exit	6	50	0	0	0	1	exit	6	50	0	0	0	1	Pass

Test#27														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(100)	idle	1	100	0	0	0	0	idle	1	100	0	0	0	0	Pass
insert_small_cups(2)	idle	1	100	0	2	0	0	idle	1	100	0	2	0	0	Pass
insert_large_cups(2)	idle	1	100	2	2	0	0	idle	1	100	2	2	0	0	Pass
coin()	idle	1	100	2	2	25	0	idle	1	100	2	2	25	0	Pass
coin()	idle	1	100	2	2	50	0	idle	1	100	2	2	50	0	Pass
coin()	idle	1	100	2	2	75	0	idle	1	100	2	2	75	0	Pass
coin()	coins inserted	2	100	2	2	0	0	coins inserted	2	100	2	2	0	0	Pass
sugar()	sugar	3	100	2	2	0	0	sugar	3	100	2	2	0	0	Pass
tea()	sugar	3	100	2	2	0	0	sugar	3	100	2	2	0	0	Pass
small_cup()	sugar	3	100	2	2	0	2	sugar	3	100	2	2	0	2	Pass
tea()	idle	1	100	2	1	0	2	idle	1	100	2	1	0	2	Pass
dispose()	exit	6	100	2	1	0	2	exit	6	100	2	1	0	2	Pass

Test#28														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
coin()	coins inserted	2	25	0	0	0	0	coins inserted	2	25	0	0	0	0	Pass
sugar()	sugar	3	25	0	0	0	0	sugar	3	25	0	0	0	0	Pass
tea()	sugar	3	25	0	0	0	0	sugar	3	25	0	0	0	0	Pass
small_cup()	sugar	3	25	0	0	0	2	sugar	3	25	0	0	0	2	Pass
tea()	sugar	3	25	0	0	0	2	sugar	3	25	0	0	0	2	Pass
large_cup()	sugar	3	25	0	0	0	1	sugar	3	25	0	0	0	1	Pass
tea()	sugar	3	25	0	0	0	1	sugar	3	25	0	0	0	1	Pass
cancel()	idle	1	25	0	0	0	1	idle	1	25	0	0	0	1	Pass
dispose()	exit	6	25	0	0	0	1	exit	6	25	0	0	0	1	Pass

Test#29														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(1)	idle	1	25	0	1	0	0	idle	1	25	0	1	0	0	Pass
insert_large_cups(1)	idle	1	25	1	1	0	0	idle	1	25	1	1	0	0	Pass
coin()	coins inserted	2	25	1	1	0	0	coins inserted	2	25	1	1	0	0	Pass
tea()	coins inserted	2	25	1	1	0	0	coins inserted	2	25	1	1	0	0	Pass
cancel()	idle	1	25	1	1	0	0	idle	1	25	1	1	0	0	Pass
dispose()	exit	6	25	1	1	0	0	exit	6	25	1	1	0	0	Pass

Test#30														Result
	Expected							Actual						
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(100)	idle	1	100	0	0	0	0	idle	1	100	0	0	0	0	Pass
insert_small_cups(1)	idle	1	100	0	1	0	0	idle	1	100	0	1	0	0	Pass
insert_large_cups(1)	idle	1	100	1	1	0	0	idle	1	100	1	1	0	0	Pass
coin()	idle	1	100	1	1	25	0	idle	1	100	1	1	25	0	Pass
coin()	idle	1	100	1	1	50	0	idle	1	100	1	1	50	0	Pass
coin()	idle	1	100	1	1	75	0	idle	1	100	1	1	75	0	Pass
coin()	coins inserted	2	100	1	1	0	0	coins inserted	2	100	1	1	0	0	Pass
sugar()	sugar	3	100	1	1	0	0	sugar	3	100	1	1	0	0	Pass
tea()	sugar	3	100	1	1	0	0	sugar	3	100	1	1	0	0	Pass
cancel()	idle	1	100	1	1	0	0	idle	1	100	1	1	0	0	Pass
dispose()	exit	6	100	1	1	0	0	exit	6	100	1	1	0	0	Pass

Test#31														Result
	Expected							Actual						
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(1)	idle	1	25	0	1	0	0	idle	1	25	0	1	0	0	Pass
insert_large_cups(1)	idle	1	25	1	1	0	0	idle	1	25	1	1	0	0	Pass
coin()	coins inserted	2	25	1	1	0	0	coins inserted	2	25	1	1	0	0	Pass
insert_small_cups(1)	coins inserted	2	25	1	1	0	0	coins inserted	2	25	1	1	0	0	Pass
small_cup()	coins inserted	2	25	1	1	0	2	coins inserted	2	25	1	1	0	2	Pass
tea()	no small cups	4	25	1	0	0	2	no small cups	4	25	1	0	0	2	Pass
insert_small_cups(-1)	no small cups	4	25	1	0	0	2	no small cups	4	25	1	0	0	2	Pass
insert_small_cups(1)	idle	1	25	1	1	0	2	idle	1	25	1	1	0	2	Pass
coin()	coins inserted	2	25	1	1	0	0	coins inserted	2	25	1	1	0	0	Pass
insert_large_cups(2)	coins inserted	2	25	1	1	0	0	coins inserted	2	25	1	1	0	0	Pass
large_cup()	coins inserted	2	25	1	1	0	1	coins inserted	2	25	1	1	0	1	Pass
tea()	no large_cups	5	25	0	1	0	1	no large_cups	5	25	0	1	0	1	Pass
insert_large_cups(-2)	no large_cups	5	25	0	1	0	1	no large_cups	5	25	0	1	0	1	Pass
insert_large_cups(2)	idle	1	25	2	1	0	1	idle	1	25	2	1	0	1	Pass
dispose()	exit	6	25	2	1	0	1	exit	6	25	2	1	0	1	Pass

Test#32														Result
	Expected							Actual						
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
coin()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
set_price(-10)	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
insert_large_cups(0)	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
insert_small_cups(-2)	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
small_cup()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
large_cup()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
tea()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
sugar()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
cancel()	idle	1	0	0	0	0	0	idle	1	0	0	0	0	0	Pass
dispose()	exit	6	0	0	0	0	0	exit	6	0	0	0	0	0	Pass

Test#33														Result
	Expected							Actual						
	state	x	price	k	k1	t	s	state	x	price	k	k1	t	s	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_large_cups(1)	idle	1	25	1	0	0	0	idle	1	25	1	0	0	0	Pass
coin()	coins inserted	2	25	1	0	0	0	coins inserted	2	25	1	0	0	0	Pass
large_cup()	coins inserted	2	25	1	0	0	1	coins inserted	2	25	1	0	0	1	Pass
tea()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
set_price(50)	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
small_cup()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
large_cup()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
tea()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
sugar()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
cancel()	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
dispose	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
insert_small_cups(1)	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
insert_large_cups(0)	no large_cups	5	25	0	0	0	1	no large_cups	5	25	0	0	0	1	Pass
insert_large_cups(2)	idle	1	25	2	0	0	1	idle	1	25	2	0	0	1	Pass
dispose()	exit	6	25	2	0	0	1	exit	6	25	2	0	0	1	Pass

Test#34															Result
	Expected							Actual							
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
insert_small_cups(1)	idle	1	25	0	1	0	0	idle	1	25	0	1	0	0	Pass
coin()	coins inserted	2	25	0	1	0	0	coins inserted	2	25	0	1	0	0	Pass
small_cup()	coins inserted	2	25	0	1	0	2	coins inserted	2	25	0	1	0	2	Pass
tea()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
set_price(50)	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
small_cup()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
large_cup()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
tea()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
sugar()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
cancel()	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
dispose	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
insert_large_cups(1)	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
insert_small_cups(0)	no small cups	4	25	0	0	0	2	no small cups	4	25	0	0	0	2	Pass
insert_small_cups(2)	idle	1	25	0	2	0	2	idle	1	25	0	2	0	2	Pass
dispose()	exit	6	25	0	2	0	2	exit	6	25	0	2	0	2	Pass

Test#35														Result
	Expected							Actual						
	state	**x**	**price**	**k**	**k1**	**t**	**s**	**state**	**x**	**price**	**k**	**k1**	**t**	**s**	
	start	1	0	0	0	0	0	start	1	0	0	0	0	0	Pass
set_price(25)	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
coin()	coins inserted	2	25	0	0	0	0	coins inserted	2	25	0	0	0	0	Pass
set_price(50)	coins inserted	2	25	0	0	0	0	coins inserted	2	25	0	0	0	0	Pass
dispose()	coins inserted	2	25	0	0	0	0	coins inserted	2	25	0	0	0	0	Pass
insert_large_cups(2)	coins inserted	2	25	0	0	0	0	coins inserted	2	25	0	0	0	0	Pass
insert_small_cups(2)	coins inserted	2	25	0	0	0	0	coins inserted	2	25	0	0	0	0	Pass
tea()	coins inserted	2	25	0	0	0	0	coins inserted	2	25	0	0	0	0	Pass
sugar()	sugar	3	25	0	0	0	0	sugar	3	25	0	0	0	0	Pass
set_price(50)	sugar	3	25	0	0	0	0	sugar	3	25	0	0	0	0	Pass
dispose()	sugar	3	25	0	0	0	0	sugar	3	25	0	0	0	0	Pass
insert_large_cups(2)	sugar	3	25	0	0	0	0	sugar	3	25	0	0	0	0	Pass
insert_small_cups(2)	sugar	3	25	0	0	0	0	sugar	3	25	0	0	0	0	Pass
tea()	sugar	3	25	0	0	0	0	sugar	3	25	0	0	0	0	Pass
cancel()	idle	1	25	0	0	0	0	idle	1	25	0	0	0	0	Pass
dispose()	exit	6	25	0	0	0	0	exit	6	25	0	0	0	0	Pass

6. Source Code

VendingMachine.java

```
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//////////////////
	void show_state(){
		System.out.println("");
		if(x==1){
			System.out.println("idle");
		}else if(x==2){
			System.out.println("coins inserted ");
		}else if(x==3){
			System.out.println("sugar");
		}else if(x==4){
			System.out.println("no small cups");
		}else if(x==5){
			System.out.println("no large_cups");
		}else if(x == 6){
			System.out.println("Vending Machine is shut down. Please start again.");
		}
		System.out.println("");
	}

	void show_all_variables(){

		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);
	}

	void show_x(){
		System.out.println("x="+x);
	}
	void show_price(){
		System.out.println("price="+price);
	}
	void show_k(){
		System.out.println("k="+k);
	}
	void show_k1(){
		System.out.println("k1="+k1);
	}
	void show_t(){
		System.out.println("t="+t);
	}
	void show_s(){
		System.out.println("s="+s);
```

```
	}
	void set_x(int y)
	{
		x = y;
	}
	void set_k(int y)
	{
		k = y;
	}
	void set_k1(int y)
	{
		k1 = y;
	}

	void set_t(int y)
	{
		t = y;
	}
	void set_s(int y)
	{
		s = y;
	}
}
```

VendingMachineTestDriver.java

```
import java.util.Scanner;


public class VendingMachineTestDriver {

	public static void main(String[] args) {
		int userOption = 0;

		   int ret=-1;

		  VendingMachine obj=new VendingMachine();
		  Scanner in = new Scanner(System.in);


		 while(userOption!=10){

			System.out.println("Please choose from below options");
			System.out.println("0. coin()");
			System.out.println("1. small_cup()");
			System.out.println("2. large_cup()");
			System.out.println("3. sugar()");
			System.out.println("4. tea()");
			System.out.println("5. insert_large_cups(int n)");
			System.out.println("6. insert_small_cups(int n)");
			System.out.println("7. set_price(int p)");
			System.out.println("8. cancel()");
			System.out.println("9. dispose()");

			System.out.println("");

			System.out.println("10. Quit");

			System.out.println("");

			System.out.println("********Testing Oriented Methods********");
			System.out.println("11.show_state()");
			System.out.println("12.show_all_variables()");
```

```
System.out.println("");
userOption=in.nextInt();

switch(userOption){
case 0:
       System.out.println("coin()");
       ret=obj.coin();
       System.out.println("The return value="+ret);
       obj.show_t();
       obj.show_x();
       break;
case 1:
       System.out.println("small_cup()");
       ret=obj.small_cup();
       System.out.println("The return value="+ret);
       obj.show_s();
       obj.show_x();
       break;
case 2:
       System.out.println("large_cup()");
       ret=obj.large_cup();
       System.out.println("The return value="+ret);
       obj.show_s();
       obj.show_x();
       break;
case 3:
       System.out.println("sugar()");
       ret=obj.sugar();
       System.out.println("The return value="+ret);
       obj.show_x();
       break;
case 4:
       System.out.println("tea()");
       ret=obj.tea();
       System.out.println("The return value="+ret);
       obj.show_x();
       break;
case 5:
       int large_cups;
       System.out.println("insert_large_cups(int n)");
       System.out.println("Enter value of n:");
       large_cups=in.nextInt();
       ret=obj.insert_large_cups(large_cups);
       System.out.println("The return value="+ret);
       obj.show_k();
       obj.show_x();
       break;
case 6:
       int small_cups;
       System.out.println("insert_small_cups(int n)");
       System.out.println("Enter value of n:");
       small_cups=in.nextInt();
       ret=obj.insert_small_cups(small_cups);
       System.out.println("The return value="+ret);
       obj.show_k1();
       obj.show_x();
       break;
case 7:
       int set_price;
       System.out.println("set_price(int p)");
       System.out.println("Enter value of p:");
       set_price=in.nextInt();
       ret=obj.set_price(set_price);
       System.out.println("The return value="+ret);
```

```
                        obj.show_price();
                        obj.show_x();
                        break;
                case 8:
                        System.out.println("cancel()");
                        ret=obj.cancel();
                        System.out.println("The return value="+ret);
                        obj.show_x();
                        break;
                case 9:
                        System.out.println("dispose()");
                        ret=obj.dispose();
                        System.out.println("The return value="+ret);
                        obj.show_x();
                        break;
                case 10:
                        System.out.println("Test Driver is Stopped");
                        System.exit(0);

                case 11:
                        System.out.println("show_state()");
                        obj.show_state();
                        break;
                case 12:
                        System.out.println("show_all_variables()");
                        obj.show_all_variables();
                        break;

                default:
                        {
                        System.out.println("Please choose only from the given options!!");
                        System.out.println("Else Enter 10 to quit");
                        }
                                break;
                }
                System.out.println("");
                System.out.println("");

        }


        in.close();


    }

}
```