PROJECT (a general description)

CS589; Fall 2016
Due Date: **November 30, 2016**
Late project: **50% penalty**
After **December 4, 2016** the project will not be accepted.

Object-Oriented and Model-Based Testing

The goal of this project is to test *VendingMachine* class that exhibits state behavior specified by the EFSM model. The source code of the class *VendingMachine* is provided in a separate file.

Description of the *VendingMachine* class:

The following operations are supported by the *VendingMachine* class:

```
VendingMachine();               //constructor
int coin();                     //a quarter is inserted
int small_cup();                //small cup is selected
int large_cup();                //large cup is selected
int sugar();                    //sugar button is pressed
int tea();                      //tea button is pressed
int insert_large_cups(int n);   //n large cups are inserted into the vending machine
int insert_small_cups(int n);   //n small cups are inserted into the vending machine
int set_price(int p);           //new price of a cup of tea is set
int cancel();                   //cancel selection for a cup of tea
int dispose();                  //this operation is used to terminate the operation of
                                //the vending machine
```

Unless stated differently, each method (operation) returns 1 when the operation is successfully completed; otherwise, zero (0) value is returned.

The *VendingMachine* class is a state-based class that is used to control a simple vending machine. The vending machine disposes a cup (small or large) of tea with/without sugar. The EFSM specifies the behavior of the *VendingMachine* class. The EFSM for the *VendingMachine* class is provided in a separate file. Notice that the price for all drinks is the same.

TESTING
In this project the goal is to test the provided implementation (source code) of the *VendingMachine* class. In order to test the *VendingMachine* class, you are supposed to implement a testing environment that should contain a class test driver to execute test cases. The following testing methods should be used:

1. Model-Based Testing. Use the provided EFSM model to test the *VendingMachine* class. Design test cases for the *VendingMachine* class so that all 2-transition sequences testing criterion (all transition-pairs) is satisfied based on the provided EFSM, i.e., all 2-transition sequences are exercised during testing.
2. Design a set of additional test cases so each default (ghost) transition in the EFSM is tested.
3. Use multiple-condition testing to design additional test cases to test predicates of conditional-statements in operations/methods. Notice that if a predicate contains only a simple condition, the multiple-condition testing is equivalent to the branch testing for this predicate.
4. Execute all test cases designed in steps 1, 2, and 3. For each test case, determine the correctness/incorrectness of the test results. If for a given test case the results are incorrect (test failed), identify the cause of incorrectness (a defect) in the source code of the *VendingMachine* class.

In the testing environment, you need to introduce testing-oriented methods (in the *VendingMachine* class) that will be used to watch "internal states" of the *VendingMachine* object in order to determine the correctness/incorrectness of the results for test cases.

Note: As a tester, you are not supposed to modify the logic (source code) of any operation/method of the *VendingMachine* class. In addition, notice that the source code under test may contain defects.

Sample test case:
Test #1: set_price(25), insert_large_cups(5), coin(), large_cup(), tea(), dispose()

Notice when the EFSM model is "executed" on this test (sequence of events), the following sequence of transitions are traversed: T_1, T_4, T_2, T_7, T_{19}, T_{12}, T_5

The detailed description for the project report and deliverables will be presented later on.