HOMEWORK ASSIGNMENT #2

CS589; Fall 2022
Due Date: **October 20, 2022**
Late homework 50% off
After **October 25** the homework assignment will not be accepted.

This is an **individual** assignment. **Identical or similar** solutions will be penalized.

Submission: All homework assignments must be submitted on the Blackboard. The submission **must** be as one PDF-file (otherwise, 10% penalty will be applied).

Consider the following source code of function *F()*:

```
1:      int F(int a, int b, int c)
        { int type, t, i;
2:              type = 0;
3:              i=0;
4:              while (i<=1) {
5:                      if (a >= c) {
6:                              t = a;
7:                              a = c;
8:                              c = t;
                        }
9:                      if (b >= c) {
10:                             t = b;
11:                             b = c;
12:                             c = t;
                        }
13:                     if (a + b <= c) {
14:                             type = -1;
                        }
15:                     if (type >= 0) {
16:                             if ((a == c) || (b == c)) {
17:                                     type = type+1;
                                }
18:                             if ((type==0)&&(a == b)) {
19:                                     type = type+1;
                                }
20:                             if (type>=0)
21:                                     if ((a*a+b*b==c*c) || (a*a+c*c==b*b))
22:                                             type=type+1;
                        }
23:                     i=i+1;
                }
24:             return type;
}
```

PROBLEM #1 (35 points): Branch and Multiple-Condition testing.

- For function *F()* derive a set of test cases that covers **branch testing** (all branches are executed). Show that your test cases execute all branches, i.e., for each branch show which test executes this branch.
- Derive additional test cases that cover **multiple-condition testing**. Show that your test cases execute all combinations of simple conditions for all complex predicates, i.e., for each combination of simple conditions indicate which test "executes" this combination. Notice that there are 3 conditional statements with complex predicates.

Note: Sample test cases:
Test #1: *a*=54, *b*=72, *c*=25
Test #2: *a*=0, *b*=5, *c*=1

PROBLEM #2 (30 points): Data-flow (definition-use) testing.
For function *F()* design a set of test cases that cover **data-flow testing**:
(1) identify all data flows (definition-use pairs) for variables ***c*** and ***type***,
(2) derive a set of test cases that "execute" all identified data flows.

Show that your test cases execute all data flows (definition-use pairs), i.e., for each data flow show which test executes this data flow (definition-use pair).

PROBLEM #3 (35 points): State-based testing
The *GasPump* component supports the following operations:

Activate (int a)	// the gas pump is activated where *a* represents the price of the gas
Start()	//start the transaction
Credit()	// pay for gas by a credit card
Reject()	// credit card is rejected
Cancel()	// cancel the transaction
Approved()	// credit card is approved
Cash(int c)	// pay for gas by cash, where *c* represents prepaid cash
StartPump()	// start pumping gas
PumpGallon()	// one gallon of gas is disposed
Stop()	// stop pumping gas

The *GasPump* component is a state-based component that is used to control a simple gas pump. Users can pay by cash or a credit card. The gas pump disposes the gasoline. The price of gasoline is provided when the gas pump is activated.
A simplified EFSM model for the *GasPump* component is shown on the next page in Figure 1.

a. Design a set of test cases that satisfy the **transition-pairing testing** criterion. For each state identify all transition pairs. Design a set of tets cases that "cover" all identified transition pairs. For each transition-pair indicate which test executes this transition-pair.
b. Design a set of test cases that satisfy the **default-transition testing** criterion. Identify all default transitions in each state. Design a set of tets cases that "execute" all identified default transitions. For each default transition indicate which test executes this default transition.

Sample test cases:
Test #1: Activate(5), Start(), Credit(), Approved(), StartPump(), PumpGallon(), Stop()
Test #2: Activate(4), Start(), Cash(5), StartPump(), PumpGallon(), PumpGallon()
Notice that the following transitions are executed/traversed on these two tests:
Test #1: T_1, T_2, T_3, T_6, T_{10}, T_8, T_{11}
Test #2: T_1, T_2, T_5, T_{10}, T_8, T_9

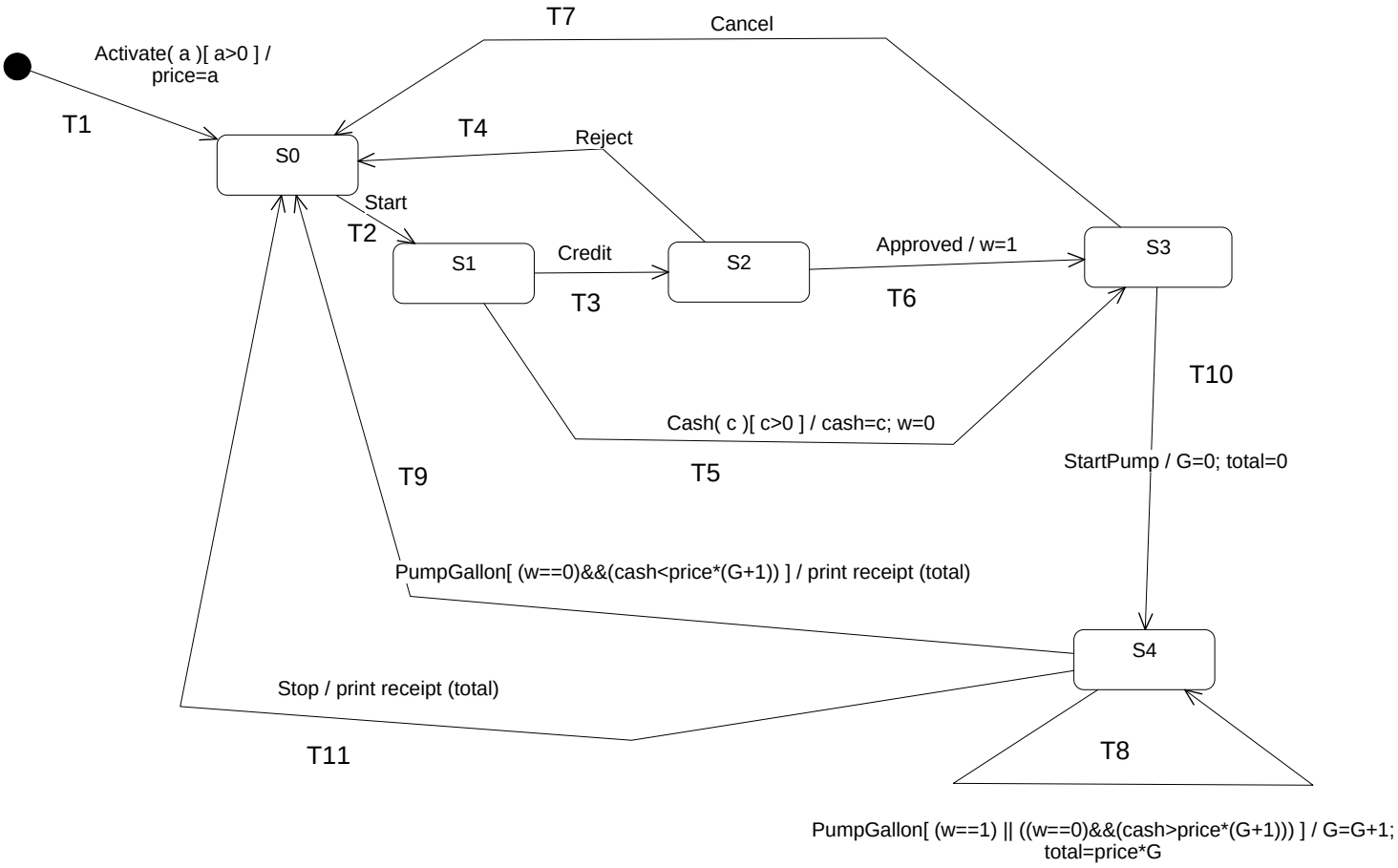

Figure 1: EFSM for GasPump component