HOMEWORK ASSIGNMENT #2

CS589; Fall 2014
Due Date: **October 22, 2014**
Late homework 50% off
After **October 26** the homework assignment will not be accepted.

This is an **individual** assignment. **Identical or similiar** solutions will penalized. The **hardcopy** of the assignment must be submitted. Electronic submissions are not acceptable. Notice that the Blackboard homework assignment submissions are only considered as a proof of submission on time (before the deadline). If the hardcopy is different than the electronic version submitted on the Blackboard, then **50% penalty** will be applied.

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

```
1:      int triangle_type(float  x1, float y1, float  x2, float  y2, float  x3, float  y3)
{       int type;
        float a, b, c, t;
2:      type=0;
3:      a=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
4:      b=sqrt((x1-x3)*(x1-x3)+(y1-y3)*(y1-y3));
5:      c=sqrt((x3-x2)*(x3-x2)+(y3-y2)*(y3-y2));
6:      if (a > b) {
7:              t = a;
8:              a = b;
9:              b = t;
        }
10:     if (a > c) {
11:             t = a;
12:             a = c;
13:             c = t;
        }
14:     if (b > c) {
15:             t = b;
16:             b = c;
17:             c = t;
        }
18:     if (a + b <= c) {
19:             type = -1; // NOT_A_TRIANGLE;
20:     }
        else {
21:             type = 3; //SCALENE;
22:             if ((a == b) && (b == c)) {
23:                     type = 2; //EQUILATERAL;
                }
24:             else if ((a == b) || (b == c)) {
25:                     type = 1; // ISOSCELES;
                }
26:             if (type==3)
27:                     if ((a*a+b*b==c*c) || (a*a+c*c==b*b) || (b*b+c*c==a*a))
28:                             type=type+1; //SCALENE && RIGHT
        }
29:     return type;
}
```

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

- For function *triangle_type()* 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: *x1*=54, *y1*=72, *x2*=25, *y2*=72, *x3*=7, *y3*=17
Test #2: *x1*=0, *y1*=0, *x2*=1, *y2*=0, *x3*=0, *y3*=1

PROBLEM #2 (35 points): Data-flow (definition-use) testing.
For function *triangle_type()* design a set of test cases that cover data-flow testing:

(1) identify all data flows (definition-use pairs), and then
(2) derive a set of test cases that "cover" all 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 (30 points): State-based testing
The **ATM** component supports the following operations:

create()	// ATM is created
card (int x, string y)	// ATM card is inserted where *x* is a balance and *y* is a pin #
pin (string x)	// provides pin #
deposit (int d);	// deposit amount *d*
withdraw (int w);	// withdraw amount *w*
balance ();	// display the current balance
lock(string x)	// lock the ATM, where x is a pin #
unlock(string x)	// unlock the ATM, where x is pin #
exit()	// exit from the ATM

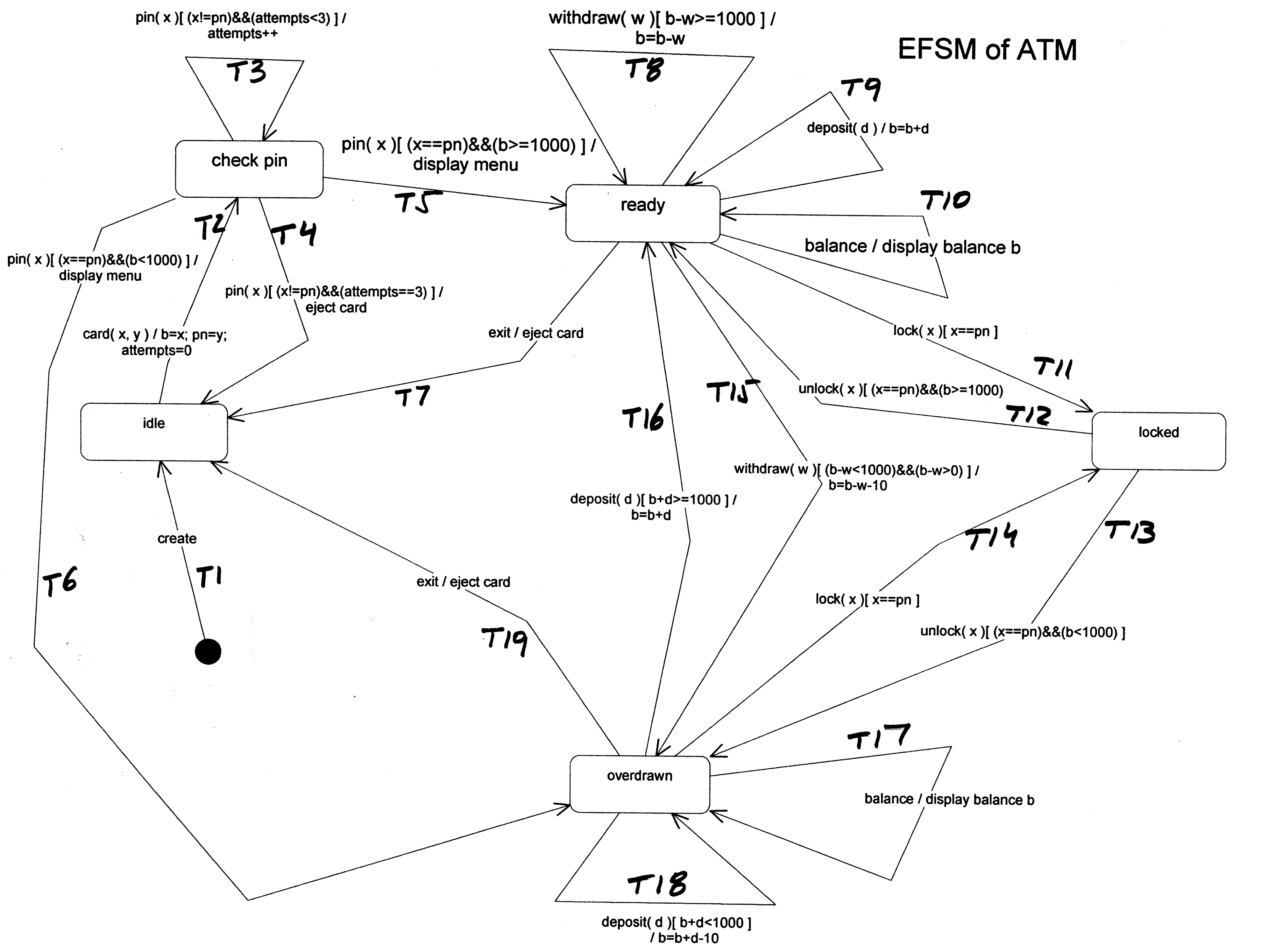
A simplified EFSM model for the ATM component is shown below:

a. Design a set of test cases so each transition is "covered" in the EFSM diagram. For each test case show which transitions are "covered".
b. Design additional test cases that satisfy the transition-pairing testing criterion. Show that your test cases "execute" all transition-pairs, i.e., for each transition-pair indicate which test executes this transition-pair.

Sample test cases:
Test #1: create(), card(2000,"abc"), pin("abc"), deposit(20), exit()
Test #2: create(), card(100,"abc"), pin("xyz"), pin("abc"), deposit(20), exit()

Notice that the following transitions are executed/traversed on these two tests:
Test #1: T1, T2, T5, T9, T7
Test #2: T1, T2, T3, T6, T18, T19

EFSM of ATM


pin(x)[(x!=pn)&&(attempts<3)] / attempts++
T3
withdraw(w)[b-w>=1000] / b=b-w
T8
T9
deposit(d) / b=b+d
check pin
pin(x)[(x==pn)&&(b>=1000)] / display menu
T5
ready
T10
balance / display balance b
T2
T4
pin(x)[(x==pn)&&(b<1000)] / display menu
pin(x)[(x!=pn)&&(attempts==3)] / eject card
card(x, y) / b=x; pn=y; attempts=0
exit / eject card
lock(x)[x==pn]
T11
T7
T15
unlock(x)[(x==pn)&&(b>=1000)
T12
idle
T16
locked
withdraw(w)[(b-w<1000)&&(b-w>0)] / b=b-w-10
deposit(d)[b+d>=1000] / b=b+d
create
T14
T13
T6
T1
exit / eject card
lock(x)[x==pn]
T19
unlock(x)[(x==pn)&&(b<1000)]
T17
overdrawn
balance / display balance b
T18
deposit(d)[b+d<1000] / b=b+d-10