HOMEWORK ASSIGNMENT #2

CS589; Fall 2013
Due Date: **October 16, 2013**
Late homework 50% off
After **October 20** the homework assignment will not be accepted.
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).

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

```
1:  int trapezoid_type(int a, int b, int c, int d) {
        int type;
        float h;
2:      type=0;
3:      h=0;
4,5:    if (a==b) type=type-1 ;                  // incorrect input
6,7:    if ((a<=0) || (b<=0)) type= type -1 ;    // incorrect input
8,9:    if ((c<=0) || (d<=0)) type= type -1 ;    // incorrect input
10:     if (type>=0) {
11,12:    if (a!=b) h=sqrt((-a+b+c+d)*(a-b+c+d)*(a-b+c-d)*(a-b-c+d))/(2*abs(b-a));
13:           if (h*h>0) {
14:               type=type+1;                          //trapezoid
15,16:            if ((c==h) || (d==h)) type=type+3; //right trapezoid
17,18:            if (c==d) type=type+2;                //isosceles trapezoid
19:               if (type!=3)
20:                   if ((a!=c)&&(a!=d))
21,22:                        if ((b!=c)&&(b!=d)) type=type+1;     //scalene trapezoid
              }
        }
23:     return type;
    }
```

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

- For function *trapezoid_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 5 conditional statements with complex predicates.

Note: Sample test cases:
Test #1: *a*=5, *b*=7, *c*=4, d=5
Test #2: *a*=7, *b*=21, *c*=25, d=10

PROBLEM #2 (35 points): Data-flow (definition-use) testing.
For function *trapezoid_type()* design a set of test cases that covers 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 Vending_Machine component (class) supports the following operations:

void create()
void coin ()
void insert_cups(int k)
void coffee()
void cream ()
void cancel()

A simplified EFSM model for the Vending_Machine 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(), insert_cups(5), coin(), coin(), coffee()
Test #2: create(), coin(), insert_cups(1), coin(), cream(), coffee(), insert_cups(10)
Notice that the following transitions are executed/traversed on these two tests:
Test #1: T1, T3, T4, T8, T5
Test #2: T1, T2, T3, T4, T15, T10, T9