HOMEWORK ASSIGNMENT #1 (modified)

CS589; Fall 2013
Due Date: **September 18, 2013**
Late homework 50% off
After **September 22** 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).

SPECIFICATION-BASED TESTING

Suppose a software component (called a Car Insurance System) has been implemented to handle processing the annual renewal of a hypothetical auto insurance policy. The following are requirements for the component:

> If the insured made one claim in the last year and is age 24 or older, the increase is $50 and no letter is sent. If the insured had no claims in the last year and is age 23 or younger, the increase is $75 and no letter is sent. If the insured had no claims in the last year and is age 24 or older, the increase is $35 and no letter is sent. If the insured made two, three, or four claims in the last year and is age 23 or younger, the increase is $400 and a warning letter is sent. If the insured made one claim in the last year and is age 23 or younger, the increase is $150 and a warning letter is sent. If the insured made two, three, or four claims in the last year and is age 24 or older, the increase is $200 and a warning letter is sent. If the insured made five or more claims in the last year, the policy is cancelled. If a car is 10 (or more) years old, 10% reduction to the increase is applied.

The component accepts the input in the following format (5 input variables):

last name, person age, car type, car age, # of claims

Assumptions:

- Assume that the *last name* contains only correct (valid) characters.
- The maximum size of the "*last name*" is 20 characters.
- Assume *person age* is an integer. The minimum *age* is 18 and the maximum *age* is 120.
- The *car type* can be: *sedan, mini-van*, *truck*, *SUV*
- Assume the maximum *# of claims* is 12.
- *car age* represents the age of the car. Assume that *car age* is an integer. The minimum *car age* is 1 and the maximum *car age* is 120.

A sample test case:
Test #1:

last name:	Smith
person age:	27
car type:	truck
car age:	12
# of claims:	3

PROBLEM #1 (35 points): Equivalence partition testing
Identify input conditions for the Car Insurance System. From the identified input conditions list equivalence valid and invalid sub-domains (classes). Based on identified sub-domains design test cases using:

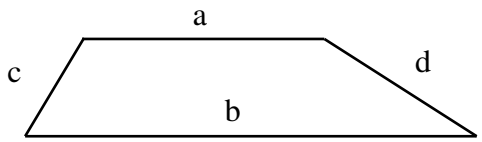
a. Strong normal equivalence testing,
b. Weak robust equivalence testing

Hint: Before designing test cases, identify related/unrelated input conditions.

PROBLEM #2 (30 points): Boundary-value testing
Based on identified sub-domains in Problem #1 design:

1. Boundary-Value Analysis test cases.
2. Robustness test cases.

PROBLEM #3 (35 points): Decision-Table based testing
A *Trapezoid* program accepts four integer numbers *a*, *b*, *c*, and *d* as input. These inputs represent sides of a trapezoid as shown below:



Notice that a trapezoid is a quadrilateral with only one pair of parallel sides (*a* and *b*). For more information look at: http://en.wikipedia.org/wiki/Trapezoid

The output to the program is the type of trapezoid determined by four sides:

- trapezoid
- right trapezoid
- isosceles trapezoid (non-parallel sides are equal)
- scalene trapezoid (no equal sides)
- not trapezoid
- invalid input (for inputs that violate input conditions)

The following input conditions are identified for four inputs *a, b, c, d*:

1. *a, b, c, d* are *integer* numbers
2. $0 < a, b, c, d \leq 1{,}000$
3. $b \neq a$

Use decision-table based testing to design test cases to test the *Trapezoid* program. Provide a decision table and test cases derived from the decision table.

A sample test case for this program:
Test #1: *a*=54, *b*=72, *c*=25, d=27

Note: In your solution conditions cannot be complex logical expressions, e.g.,

(b=a) and (c=d)

is **not acceptable** as a condition in the decision table.
The conditions in a decision table must simple conditions, e.g., *(b=a)* is an acceptable condition; *(c=d)* is also an acceptable condition.