HOMEWORK ASSIGNMENT #1

CS589; Fall 2022
Due Date: **September 23, 2022**
Late homework 50% off
After **September 27,** 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).

SPECIFICATION-BASED TESTING

Suppose a software component (called a Grader component) has been implemented to automatically compute a grade in a course. A course taught at a university has two exams and a project. To pass the course with grade C a student must score at least 45 points in the Exam-1, 50 points in Exam-2, and 50 points in Project. Students pass the course with grade B if they score at least 60 points in the Exam-1, 55 points in Exam-2, and 60 points in Project. If, in addition to this, the average of the exams is at least 80 points and they score at least 70 points in Project then students are awarded a grade A. Final grades for the course are: A, B, C, and E. The Grader component accepts six inputs:

Last name
First name
Student #
Exam-1
Exam-2
Project

Assumptions:

- Assume *Exam-1, Exam-2, Project* are integers.
- The ranges for the exam scores are:
  - $0 \leq Exam\text{-}1 \leq 90$
  - $0 \leq Exam\text{-}2 \leq 100$
  - $0 \leq Project \leq 80$

- The maximum size of the "*First name*" is 15 characters and "*Last name*" is 20 characters.
- *Student #* is a number represented as a 9-character string in the following format: AXXXXXXXX, where X is a digit.

Sample test case for the Grader component:

Test #1: *Last name*=Smith*, First name*=John*, Student #*=A11112222*, Exam-1*=57, *Exam-2* = 64, *Project* = 75

PROBLEM #1 (35 points): Equivalence partition testing
Identify input conditions for the Grader component related to:

1. Last name
2. First name
3. Student #
4. Exam-1
5. Exam-2
6. Project

From the identified input conditions list equivalence valid and invalid sub-domains (classes). Based on the 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 the identified sub-domains in Problem #1 design:

1. Normal Boundary-Value Analysis test cases.
2. Robust Boundary Value test cases.

PROBLEM #3 (35 points): Decision-Table based testing

Suppose a software component COMMISION has been implemented to automatically compute a commission for salespersons in XYZ-store. If a non-salaried salesperson sells an item that is neither standard nor bonus to someone other than a regular customer, he/she receives a 7% commission, unless the item costs more than $5,000, in which case the commission is 3%. For all salespeople, if a standard item is sold, or if any item is sold to a regular customer, no commission is given. If a salaried salesperson sells a bonus item, he/she receives a 3% commission, unless the item sells for more than $500, in which case he/she receives a flat $30 commission. If a non-salaried salesman sells a bonus item to someone other than a regular customer, he/she receives a 7% commission, unless the item sells for more than $500, in which case he/she receives a flat commission of $65. In all other cases, a salesman receives a 2% commission.

The component accepts five inputs:

Salesman name
Salesman type
Item type
Customer type
Item price

Assumptions:

- The maximum size of *Salesman name* is 20 characters
- *Salesman type* values: Salaried (S), Non-salaried (NS)
- *Item type* values: Standard (ST), Bonus (B), General (G)
- *Customer type* values: Regular (R), Walk-in (W)
- *Item price* is an integer

Sample test cases for the component:

Test #1: *Salesman name=Smith, Salesman type=NS, Item type=ST, Customer type=W, Item price=1400*

Test #2: *Salesman name=Brown, Salesman type=S, Item type=B, Customer type=R, Item price=200*

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

Note: In your solution conditions cannot be complex logical expressions. For example:

(*Item price* ≤ 500) *and (Item type=B)*

is **not acceptable** as a condition in the decision table. However, "*Item price* ≤ 500" is a condition; "*Item type=B*" is a condition.