This Assignment continues with the consideration of classes and objects, extending to class hierarchies (Chapter 11). These provide efficient ways to represent types of real-world classes along with sub-classes inheriting attributes and methods, but also with attributes and methods of their own.

The rectangle program is useful for interaction games, particularly shooter games, which commonly test among two geometric objects (*e.g.*, projectile and target) whether one is contained in the other, they overlap or abut or are distinct.  A convenient way to do that is to surround the object with the smallest rectangle that can contain it since the relative positions of two rectangles is easy to test.

Respond to the following exercises in the suggested order:

|  |  |  |
| --- | --- | --- |
| Assignment 6 | 11th Edition | 10th Edition |
| CS | 10.13, 11.1 w/ 12.5 illegal triangle exception | 10.13, 11.1 w/ 12.5 illegal triangle exception |
| MIS | 9.7/11.8 | 9.7/11.8 |

Input to 10.13 needs to check for negative sides and input to 11.1 should validate the triangle for having sides that actually form a triangle, that is, the sum of any two sides must be greater than the third.

For the MIS exercises the inputs to 9.7/11.8 should be validated.  In a brief prompt indicate what the valid data rules are for all inputs that could be entered but are not valid, like negative interest rate.

Rectangle 2D program 10.13, should consist of three loops, the outer repeat loop, a middle loop to enter a**base rectangle**, and an inner loop for **test rectangles** to be checked for inclusion, overlap, abutment, and distinctions to the base rectangle. The program should be structured so that the use enters the first rectangle which should not have to be entered again. Every time the program is run with new test rectangles the new shapes should be compared to the same base rectangle entered first. The width and height of all rectangles should be validated to be positive.  Test should be done for contain, overlap, abut, and distinct rectangles.  All 4 tests should be done automatically for every test rectangle, **do not ask the user which tests to make**.  Few games ask the user if they should report these interactions.  The program proceeds appropriately by showing their effects.

For the MIS flavored program, regardless of what it says in the account class problems, all the objects entered should have an ID as an attribute, usually an employee or student university ID other than a SSN.

Structure the programs into object classes, like rectangles and accounts, and test classes, each in their own files.  For the CS flavored exercises there should be 4 or 5 flies, two for the test classes, one each for rectangles and triangles, and possibly a GeometricObject class which is extended to create the rectangle and triangle classes.

The account classes should also have at least a test class and an account class in separate files but may be structured inot more files for other classes.  This is a common design problem for hierarchical class structures.

Remember:

1. All programs need a brief introduction printed for the user saying what the program will do.
2. The prompts to the user for the inputs required for the program need to say what each input is (*,*letter, word, number phrase) and how it should be formatted (*e.g.,* 3 number separated by spaces).
3. Write the program with a repeat loop so that when the first set of input is done with the programs ask the user if it should run again for more, presumably different input. There should be an option to exit or to repeat the program with new input.