UNIVERSITY OF MICHIGAN

EECS 282

Assignment 2 – Defensive Programming I (25 points). Extra credit: 5 points

---

Due Date: **Thursday, January 21st at 11:55 PM**. The deadline is precise and the time is decided by ctools. You would be using late days for submission if ctools records the receipt as after the above time. We recommend submission a few hours before the above deadline to allow for slow response time from ctools. You can submit multiple times. Ctools only saves the latest submission.

Note that the problems on this assignment do not necessarily require you to write programs, unless you wish to verify or check your answer. Please submit the solutions as a plain-text files. The java files are not required, except for Problems 1 and 2. Note: This final version includes an additional problem was (extra credit) that was not in the draft version. The deadline for this homework is also shifted to Thursday.

Problem 1: (10 points)

Following is pseudo-code that decides whether a particular day is a weekday or a weekend day. Assume that 1 corresponds to Monday, 2 to Tuesday, …, 7 to Sunday. Write it in two ways in Java:

(a) Using if..else if .. else statements.

(b) Using switch statements.

```
int day;
day =  …. ;   // Put In your test value here.
System.out.println("the value of day is " + day);
if day is Monday:
        print day, " is a weekday";
else if day is Tuesday:
         print day, "is a weekday";
…
else if day is Saturday:
         print day, "is a weekend day";
else:
        print day, " is a weekend day";
```

When you come up with an answer, revisit your code and make sure that you eliminate duplicate code by combining conditionals or by sharing among the various case statements by eliminating break statements. Also, handle possible errors in the value of day (use *assert* for handling errors).   Submit your solution to both parts (a) and (b) as Problem1a.java and Problem1b.java. Include screenshots from the runs for day = 4 and day = 9.

Problem 2: (8 points)

(a) Consider the following code segment that uses a break statement. Rewrite it so that it does the same thing and without additional loop iterations, but without using a break or continue statement.

```
int x = 0;
int sum = 0;
int N = …;     // some positive value.
int max = …; // some positive value.
for  (x = 0; x < N; x++) {
            sum = sum + x*x;  // ** Is not valid in Java. It is valid in C++ and Python.
            if (sum > max)  break;
            System.out.println("x is " + x);
}
System.out.println("x is "  + x);
```

For this part, submit your solution in a working Java program named Problem2.java. Use the values N = 1000 and max = 50000 within your code. We should be able to compile and run your code. Include a snapshot of a run.

Problem 3: 5 points (no code submission required, but you may want to test your fix)

Java has arrays, which are basically fixed-size lists of a single type of value. For example,

```
int[] m =  new int[100];
```

declares an array m, which contains 100 integer values. Java initializes all the values to 0. The elements of the array can be accessed as m[0], m[1], …., m[99]. Accessing the array element using an index that is negative or too large, e.g., m[-1] or m[100], generates an exception (error) in Java because the index must be between 0 and 99.

A student wrote the following code to permit a user to provide a start value and a count, and print out count values, starting from index start.

```
int start;
int count;
… read the values of start and count by asking the user. Code omitted. …

if (start >= 0 && count >= 0 && start + count < 100) {
            for (int i = 0; i < count; i++)  System.out.println(m[start + i]);
```

Unfortunately, he found that a user could provide bad values to start and count that would cause the code to give an array index-out-of-bound exception.

(a) Give sample values of start and count that could induce the array index out-of-bound exception.

(b) Give the fix to the code by modifying the *if condition* so that the exception would not occur, but it would work correctly when start and count are in proper range. Note that it is possible to fix the code by adding just one additional comparison in the *If condition*. You can add up to two additional comparison operations.

Problem 4: (2 points) (no code submission required)

Consider the following code:

```
float x = …;  // some value here. Omitted.
if (x > 0) System.out.println("x is positive");
else if (x <= 0) System.out.println("x is 0 or negative");
else assert(false);
```

A student was surprised to see that when the code was run, the assert statement executed. Give a value of x for which the assertion could fail in Java.

Problem 5: (5 points) - Extra Credit. Thinking problem.

Consider the code in BallGame.java. This code simulates a game that is played as follows. You put in some unknown number of red and blue balls in a bag. You then take out two balls at random from the bag.

- If they are both the same color, you toss the two balls aside and put in a blue ball back in the bag.
- If they are different colors, you toss the blue one aside and put in the red ball back in the bag.

You keep repeating the above process till only one ball is left. Note that the # of balls reduces by one at every round, so eventually, you will end up with one ball.

If the final color of the ball is red, Reds win. Else Blues win.

Question: It turns out that you can precisely determine whether the Reds win or Blues win, based on the initial color counts. Think about it and state how. You can run BallGame.java repeatedly to hope to discover the relationship between the winner and the initial counts. But, then, you must explain why the relationship holds. You can give a loop invariant (property) that proves your answer. **Hint:** think about how the count of red and blue balls changes every round of the loop. Is there a relationship that holds between old and new counts after every loop round?

How to submit:

Submit the assignment on ctools. For problems 3, 4, and 5, submit plain-text files.

For questions 1 and 2, please submit *.java files and a screenshot of the running programs, as specified in the problems.