Homework Assignment 5 (130 Points)
CSE 464, Fall 2025
CIDSE, Arizona State University
Due October 18th by 11:59 pm

Objective: To enhance knowledge about code-based testing, control flow graph, Junit testing based on control flow graphs, and Data path testing

Prerequisite: You need to have Junit installed and configured to run in your IDE. Preferably, Eclipse. Review Module 7 lectures and examples posted. Also, Junit tutorials are posted in the Tutorials module

Submission: submit the PDF version of the completed "HW_4 SubmissionTemplate" document.

Q1. [50 Points] The following function is supposed to take two strings and the length as parameters and returns number of words in the two strings that are of the same length and similar (ignore uppercase and lowercase).

```
// Method to count similar words of given length

public static int countSimilarWords(String str1, String str2, int length)

{
   // Split words by whitespace and punctuation
   String[] words1 = str1.split("\\W+");
   String[] words2 = str2.split("\\W+");

   Set<String> set1 = new HashSet<>();

   for (int i = 0; i < words1.length; i++) {
       String w = words1[i];
       if (w.length() == length) {
           set1.add(w.toLowerCase());
       }
   }

   int count = 0;

   for (int j = 0; j < words2.length; j++) {
       String w = words2[j];
       if (w.length() == length && set1.contains(w.toLowerCase())) {
           count++;
       }
   }
   return count;
}
```

a) [20 Points] Draw the control-flow graph, identify independent paths and design test cases to perform basic path testing.

Note: You can hand-draw the control flow graph scan and submit the PDF version.

b) [10 Points] Implement Junit code to run test cases for each path. Upload the Junit test cases into your google drive and include the link in HW_4 SubmissionTemplate document.

c) [20 Points] Implement Parameterized Junit test case file that tests `countSimilarWords` method using parameterized Junit testing. Also review lecture and the example given that discussed parameterized testing. Upload the Junit test cases into your google drive and include the link in HW_4 SubmissionTemplate docuument.

2.[50 Points] Control Path Testing: Credit Card Verification Problem

Consider the following program that checks if a credit number is a valid card number.

The last digit of a credit card number is the check digit, which protects againts transaction errors. The following method is used to veryfy credit card numbers. Follwing steps explains the algorithm in determining if a credit card number is a valid card.

- Starting from the right most digit, form the sum of every other digit. For example, if the credit card is number is 3125145643589795 then you form the sum 5+7+8+3+6+4+5+1 = 39
- Double each digit that we have not included in the preceding step. Add all digits of resulting numbers. For example, with the number given above, doubling the digits starting with next to last one, yields 18, 18, 10, 8, 10, 2, 4,6. Adding all digits in these values yield 1+8+1+8+1+0+8+1+0+2+4+6 = 40
- Add sum of the two preceding steps. If the last digit of the result is zero, then the number is valid number

The program asks the user 16-digit credit card number and the printout if the credit card is valid or invalid card

a) [20 Points] Draw the control-flow graph, independent paths and design test cases to perform basic path testing

Note: You can hand-draw the control flow graph scan and submit the PDF version.

b) [15 Points] Implement Junit test cases to run test cases for each path. Upload the Junit test cases into your google drive and include the link in HW_4 SubmissionTemplate docuument.

c) [15 Points] From Junit test cases above, identify two major bugs related to sum1, sum2 and FinalSum calculations in the program and then fix them

Final Submission Instructions

Submit the PDF version of the completed "HW_4 SubmissionTemplate" document that contains answers to all the question in Q1 and Q2