CS 458 - Homework 11

Deadlines

Problem 1 was completed during CS 458 Week 13 Lab.

Problems 2 onward are due by 11:59 pm on Friday, December 1, 2017

Purpose

In honor of Jalote Chapter 8 and to practice some more with JUnit, to practice writing tests for provided code; and to consider and try out examples of some of the metrics from Jalote Chapter 7.

How to submit

Problem 1 was submitted by including the required file in your team's GitHub team repository by the end of the specified lab sessions.

Submit your file for Problem 2 onward for this homework using `~st10/458submit` on **nrs-projects**, with a homework number of `11`

Problem 1

You needed to meet with your project team in a mandatory team meeting during the Week 13 Lab on Wednesday, November 15, and include a miscellaneous team meeting form `misc-meet-2017-11-15.txt` in your team GitHub repository's `sprint-x/team-meetings/misc-meetings` subdirectory, containing the required contents as described on p. 7 of the project handout.

(That is, you will be graded on working in your team during that lab, and whether your team successfully completed this file, meeting the stated specifications, during that lab.)

Problem 2

Consider the posted examples from earlier this semester, Java class `GameDie` (with source code in file `GameDie.java`) and JUnit test class `GameDieTest` (with source code in file `GameDieTest.java`).

Posted with this homework handout is `StudentGrades.java`, containing source code for a Java class `StudentGrades`.

In honor of the testing discussion in Jalote Chapter 8, and because Exam 1 showed less familiarity than expected with JUnit, write a JUnit 4.4 testing class `StudentGradesTest` in file `StudentGradesTest.java`, following the naming conventions and style used in `GameDieTest.java`.

As noted in Jalote Chapter 8, you should have to mindset of trying to "break" the software under test, and test it as thoroughly as possible, Remember that, using the DrJava IDE, if both `StudentGrades.java` and `StudentGradesTest.java` are in the same directory and you open both in DrJava, you should be able to run `StudentGradesTest`'s unit tests by clicking the "Test" button in the upper right corner.

Submit your resulting file `StudentGradesTest.java`.

FOR Problems 3 through 9

Create a file named `458hw11.txt` or `458hw11.pdf` (your choice) that starts with your name. Then give the problem number and your answer(s) for each of the remaining problems.

Consider the following two algorithms (which look to me like they're written in a version of Algol), one for linear search and one for binary search, adapted from p. 223 of the course text. You should use **these** versions **below** in your answers to the following problems.

(You should already be aware that binary search is generally more efficient in terms of execution time than linear search as the number of elements being searched increases.)

```
function lin_search(A, E): boolean
var
    i: integer;
    found: boolean;
begin
    found := false;
    i := 0;
    while (not found) and (i < A.length) do begin
        if (A[i] = E) then
            found := true;
        i := i + 1;
    end;
    lin_search := found;
end;


function bin_search(A, E): boolean
var
    low, high, mid, i, j: integer;
    found: boolean;
begin
    low := 0;
    high := A.length - 1;
    found := false;
    while (low <= high) and (not found) do begin
        mid := (low + high) / 2;
        if E < A[mid] then
            high := mid - 1
        else if E > A[mid] then
            low := mid + 1
        else
            found := true;
    end;
    bin_search := found;
end;
```

Problem 3

Determine the **cyclomatic complexity** for each of these 2 functions. (**NO** control-flow graph is required!!!)

Problem 4

See Jalote pp. 219-220 for a description of live variable complexity. Note in particular:

- A variable is considered to be **live** from its first to its last reference within a module, including all statements **between** the first and last statement where the variable is referenced.
  - note, then -- if a variable is first referenced in statement 5, and last referenced in statement 10, then it is included in the live variable count for EACH of the statements 5, 6, 7, 8, 9, and 10.
- Using this definition, the set of live variables for each statement can be computed by analysis of the module's code.
- For a statement, the number of live variables represents the degree of difficulty of the statement.
- This notion can be expanded to the entire module by defining the average number of live variables per executable statement --
  - that is, get the number of live variables per executable statement,
  - sum the number of live variables for all of the module's executable statements,
  - divide by the number of executable statements

  ...and the result is the live variable complexity for that module.

Determine and give the **live variable complexity** for each of these two functions.

- Note: I'm more interested in having you try to apply the live variable complexity algorithm as stated, to apply it consistently to both functions, and to consider how these two functions differ based on this measure, than I am in whether you make slightly different assumptions about, for example, what constitutes an executable statement.

  That said, though, here are the assumptions I am happening to make:
  - remember that we are only interested in executable statements;
  - so, I don't think the `begin` and `end` lines are executable statements,
  - nor are the function header and variable declarations;
  - that leaves `lin_search` with 7 executable statements,
  - and `bin_search` with 12 executable statements.
- ...and those are the assumptions I'll be happening to use in my example solutions to this problem.

Problem 5

Give the ratio of `lin_search`'s cyclomatic complexity over `bin_search`'s cyclomatic complexity, and give the ratio of `lin_search`'s live variable complexity over `bin_search`'s live variable complexity. Also answer this: are these two ratios similar?

Problem 6

Determine and give the number of non-comment, non-blank lines in each of these functions.

Problem 7

Consider Halstead's size metrics on Jalote pp. 215-216.

Determine and give **Halstead's length measure**:

$N = N_1 + N_2$

...where N_1 is the total number of occurrences of all of the operators in a module, and

...where N_2 is the total number of occurrences of all of the operands in a module,

for each of these functions.

- **NOTE**: Again, I'm more interested, here, in having you try to determine Halstead's measure as defined, to do so consistently for both functions, and to consider how these two functions differ based on this measure, than I am in whether you make slightly different assumptions about, for example, what constitutes an operator or operand.
  - That said, note that I am counting accessing an array element and accessing a data field as operators,
  - and taking operand to mean any expression involved in an operation (even including, then, "compound" operands),

  ...and those are the assumptions I'll be happening to use in my example solutions to this problem.

Problem 8

Still considering Halstead's size metrics on Jalote pp. 215-216, determine and give **Halstead's volume measure**:

$V = N \log_2 (n)$

...where N is Halstead's length measure that you computed in Problem 5, and

$n = n_1 + n_2$

...where n_1 is the number of distinct/unique operators in the module, and

...where n_2 is the number of distinct/unique operands in the module,

for each of these functions.

- **NOTE**: Again, I'm more interested, here, in having you try to determine Halstead's volume as defined, to do so consistently for both functions, and to consider how these two functions differ based on this measure, than I am in whether you make slightly different assumptions about, for example, what constitutes an operator or operand.
  - That said, note that I am again counting accessing an array element and accessing a data field as operators,
  - and taking operand to mean any expression involved in an operation (even including, then, "compound" operands),

  ...and those are the assumptions I'll be happening to use in my example solutions to this problem.

Problem 9

Make and give a chart showing and comparing the size measured in LOC, measured as Halstead's length measure, and measured as Halstead's volume measure, for these two functions.

Which of these measures seems more pertinent to you? AND, Why? (notice that you are being asked TWO questions here, in addition to giving the chart.)

Submit your resulting file `458hw11.txt` or `458hw11.pdf`.