**Assignment 9 (20 points), SE 421, 11/03/2021, due: Wednesday, 11/10/2020**

**Name (Last, First):**

**Submission Requirement**: Submit a zip file for your code and a PDF for the writeup.

You will submit a .zip that comprises of the following: (1) Your code (exported as a .zip just like your project code), and (2) a PDF of your writeup prepared according to requirements. Your zip must be named HW9-lastname-firstname. *Writeup*: (a) Must be typed. (b) The first page should include the top two lines with your last and the first name. The PDF should also be named HW9-lastname-firstname.

**Topic**: This homework shows how to create a *program slicing* (PS) tool. Program slicing is an important technique to analyze software for cybersecurity and safety. We also discussed its overall importance in software engineering (e.g. to set appropriate break points for systematic debugging).

**Important Point**: The implemented algorithm would work equally well on Java or C software. As we discussed in class, this is possible because eXtensible Common Software Graph (XCSG) provides a unified schema for program semantics of multiple languages.

**Intended Use of the PS tool**:

1. Select a line with a program statement by clicking on it – this is used to specify the program statement for slicing)
2. var udChains = variableSpecificDataDependence(selected,"v");

selected: specified program statement and v: specified variable from that statement.

1. show(udChains);

**Code**: You need to write the missing part for the query: variableSpecificDataDependence()

The missing part is for the logic to compute Use-Def (UD) chains.

**Resources:**

* The logic to compute the UD chains was discussed during the lectures on data flow.
* You are provided the plugin (hw9code.zip) com.se421.dataflow. This plugin operates the same way as the Path Counting project and contains a TODO in the file VariableSpecificDataDependence.java (located in the package com.se421.dataflow.queries). TODO is the missing part you need to write.
* HW9 Plugin Setup Guide, please follow it carefully to import the plug-in in eclipse and setup it.

1. (3 points) Draw by hand the complete UD graph for the function **foo** and **main**. Include these graphs in your writeup. (Hint: We have done UD graph in class for another function).
2. (4 points) Include in your writeup a hand-drawn backward program slice for:
   1. The function **foo** for the variable **t** in line 17 (DDExampleJava.java)
   2. The function **main** for the variable **d** in line 36 (DDExampleC.c)
3. (8 points) Fill in the TODO in VariableSpecificDataDependence.java for computing the UD chains. Include in your write up a copy of VariableSpecificDataDependence.java with TODO completed. Submit the complete code as a zip file.
4. (2 points) Test your code for the following:
5. The function **foo** for the variable **t** in line 17 (DDExampleJava.java)
6. The function **main** for the variable **d** in line 36 (DDExampleC.c)

Compare the results produced by the tool against the hand-drawn answers. Include in your writeup a short conclusion. The test cases are given as: DDGExampleC and DDGExampleJava. You will need to import them as Atlas projects for testing.

1. (3 points):Write a query to find all the functions f such that:
   1. There is a *call sequence* from dswrite to f and have a parameter which is a pointer to the structure type dreq

**Writeup Requirements:**

1. The hand-drawn answers for #3 and #4.
2. The code for variableSpecificDataDependence()with your part highlighted.
3. The resulting graph for test A
4. The resulting graph for test B
5. Problem 5 solution