**Assignment 4 (20 points), SE 421, 9/8/2020, due: Monday, 9/16/2020**

**Name (Last, First): Morellas, Stamatios**

**Submission Requirement**: (a) The answers should be typed. (b) The first page should include the top two lines with your last and the first name. (c) Include each question along with your answer to the question. (d) The file should be named HW4-lastname-firstname. (e) Include each question from here followed by your answer. The submission time and late penalty requirements remain the same.

**Topics**: (a) dataflow basics, (b) dataflow in presence of pointers, (c) importance of dataflow analysis for cybersecurity and safety.

**Problem 1 (5 points):** Answer the following questions based the given code**.** Answer the questions without using Atlas.

**1. int main(int a1, int a2, bool c1, bool c2, bool c3) {**

**2. int x,d,y,z;**

**3. x = a1 + a2;**

**4. d = a2;**

**5. if(c1){**

**6. x = a1;**

**7. }**

**8. if(c2){**

**9. if(c3){**

**10. y = a1;**

**11. } else{**

**12. d = d - a1;**

**13. }**

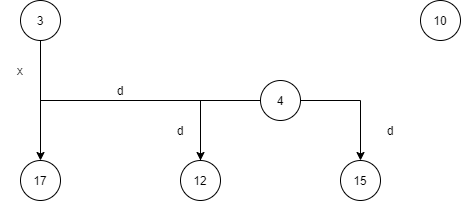
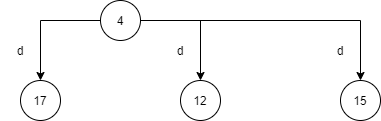
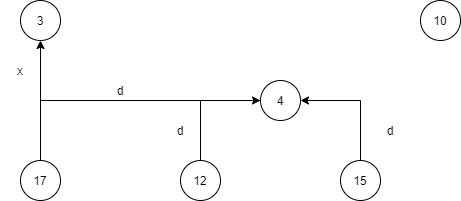
**14. } else{**

**15. d = d + 1;**

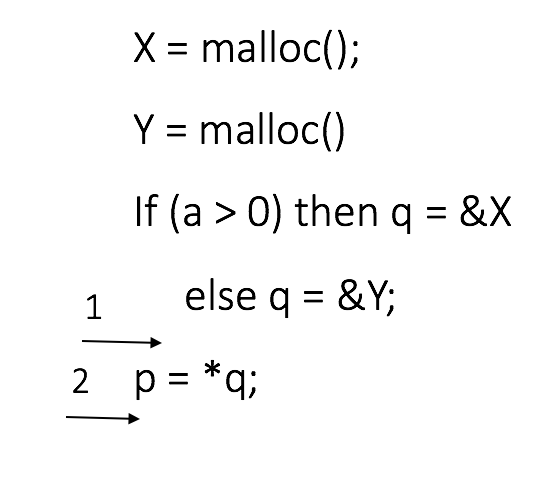
**16. }**

**17. z = x/d;**

**18. }**

1. (2 points) Draw the *DU graph*. Use statements 3, 4, 10, 12, 15, and 17 as the nodes.
   1. 
2. (1 point) Draw the *forward slice* starting with the definition of **d** at line 4.
   1. 
3. (1 point) How many *def-use* (DU) chains starting with the definition of **d** at line 4?
   1. There are **three** def-use (DU) chains starting at the statement at line 4: (4🡪17), (4🡪12), and (4🡪15)
4. (1 points) Draw the *UD graph*. Use statements 3, 4, 10, 12, 15, and 17 as the nodes.
   1. 

**Problem 2 (4 points):** Let T(p), the target of the pointer p, be the *set of objects* that p points to. If the p does not point to anything, then T(p) = null. Let O1, O2, O3 be the objects allocated by the successive **malloc** calls. Give the values of T(X), T(Y), T(p), T(q) at each of the two program points which are shown below for each of the following two code segments.

*At the Program Point 1***:**

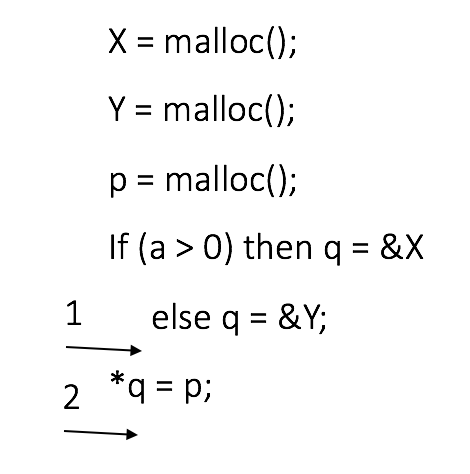
T(X) T(Y) T(p) T(q)

**{ O1 } { O2 } null { X or Y }(based on a)**

*At the Program Point 2***:**

T(X) T(Y) T(p) T(q)

**{ O1 } { O2 } { q } { X or Y }(based on a)**



*At the Program Point 1***:**

T(X) T(Y) T(p) T(q)

**{ O1 } { O2 } { O3 } { X or Y }(based on a)**

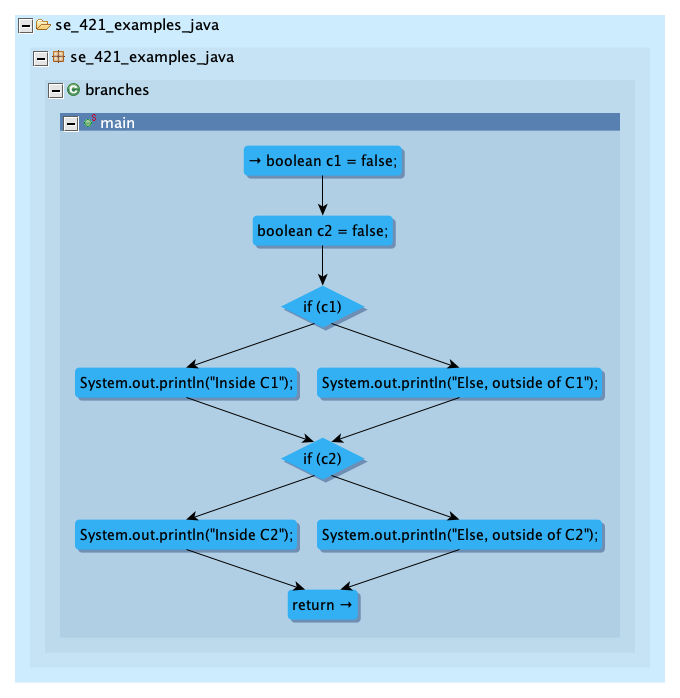
*At the Program Point 2***:**

T(X) T(Y) T(p) T(q)

**{ O1 } { O2 } { O3 } { X or Y, p }(based on a)**

**Problem 3: (2 points)** Assume that each branch is 2-way.

1. Suppose there are three non-nestedbranch statements such that each branch has one definition **V**. How many *definitions of* **V** reach the *use of* **V** right after the three branch statements?

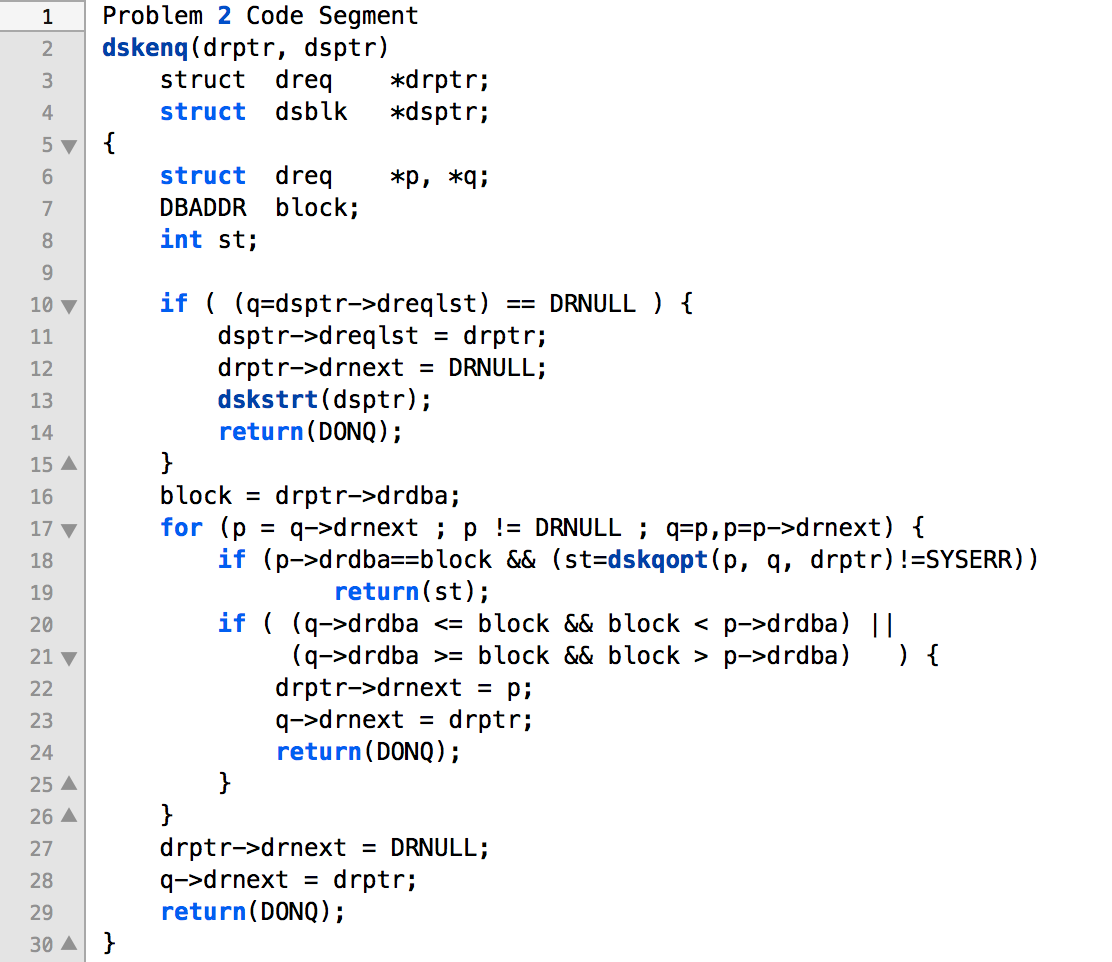


The following diagram that Ryan posted on Piazza shows an example of two two-way, non-nested branch statements. Assume that the System.out.print() statements are replaced with a definition of V (i.e. V = a, V = b, etc.). With each path in the branch containing one definition of V, this means there are two possible values that V can be defined as, even though only one will be executed when the program runs. So now if we include a third branch statement with the same format, there will be **three** definitions of V, one from each branch statement (ofthe **six** total definitions from all branch statements). The last definition of V will be the one that is used after all three branch statements are executed.

1. Suppose one definition of V is followed by three non-nested branch statements such that each branch has one use V. Assume there are no other uses of **V**. How many *uses* of V for the definition of V?

Since V is defined before the three branch statements execute and not inside each branch statement, this scenario is a little different from the previous. This means that V will be *used* one time when each branch statement is traversed, so there will be **three** uses of V for the definition of V.

**Problem 4 (7 points):** Access to the allocated memory is passed as a parameter drptr (a pointerto the allocated memory) to the function dskenq. Another parameter to dskenq is dsptr, a pointer to a global data structure. Read the code segment carefully to answer the following questions.

****

1. **(2 points)** Does dsptr provide access to the allocated memory at line 13? If yes, explain in one sentence how.
   1. **Yes**, because at line 11, a member of dsptr is being assigned the value of drptr, which is then getting updated on the line after, so dsptr should point to the allocated memory.
2. (**1 point**) Does dsptr provide access to the allocated memory at line 16?
   1. **No**.
3. (**2 points**) How many definitions of q are there in the given code? Give the line numbers for those definitions.
   1. There are **two** definitions of q in this code. One is at line 6 and the other is in the for loop at line 17, which will execute an unspecified amount of times while p != DRNULL.
4. (**2 points**) Let C1 and C2 denote the conditions for the branch statements at lines 18 and 20 respectively. Assume that the loop at line 17 does terminate. Complete the following truth table. The last column of the truth table is either YES or NO depending on whether the definition of q at line 17 reaches the use of q at line 28 along at least one control flow path.

|  |  |  |
| --- | --- | --- |
| C1 | C2 | The definition of q at line 17 reaches the use of q at line 28 – YES or NO |
| TRUE | TRUE | NO |
| TRUE | FALSE | NO |
| FALSE | TRUE | NO |
| FALSE | FALSE | YES |

**Problem 5 (2 points):** The code given in Problem 1 has the possibility of a safety vulnerability because the program would crash if d is zero on line 17. In this problem you have to show how to estimate the number of times the program must be tested to detect the *division-by-zero* vulnerability with the probability 0.9.

**Assumptions:**

1. The probability is 1/3 that during a test run a control flow path on which d can become zero would be taken.
2. A fuzzer generates random positive integers between 1 to 232 for the inputs a1, a2.
3. Integer overflows are not relevant.
4. The *division-by-zero* vulnerability occurs if (a) a control flow path on which d can become zero is taken, and (b) a1= a2.
5. What isthe probability that the vulnerability would be observed during a single test run?
   1. (*Source: Piazza forum post*)
6. What is the minimum number of tests to observe the vulnerability with probability 0.9?

Explain your answers clearly and in short. Specifically, you must write succinctly and clearly the basic probability formula(s) you have used. You are expected to find these formula(s) in case you don’t know them. Cite the source in case you searched for the formula(s).