Assignment 7 (20 points), SE 421, 10/20/2021, due: Wednesday, 10/27/2021

Name (Last, First): Hamamoto, Yuichi

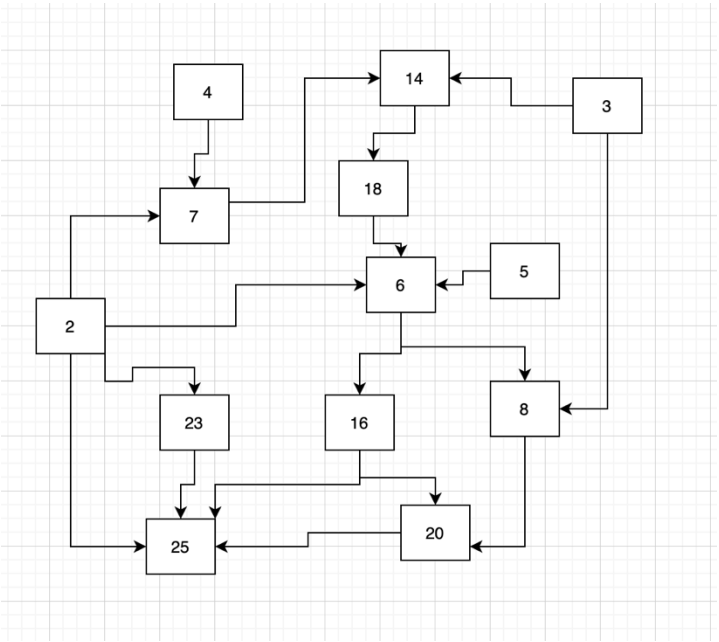
Submission Requirement: Submit a PDF named HW7-lastname-firstname. Include the top two lines with your last and the first name. Include the problem statement followed by your answer.

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

```
1. int main(bool c1, bool c2, bool c3) {
2. int a1, a2, x, d;
3. int *p1,*p2,**q1,**q2;
4. a1 = 3;
5. a2 = 5;
6. p1 = &a2;
7. p2 = &a1;
8. q2 = &p1;
9.   if(c1){
10.    *p1 = 5;
11.  } else{
12.    *p2 = 7;
13.   }
14.q1 = &p2;
15.  if(c2){
16.    x = *p1;
17.    if(c3){
18.      a2 = **q1;
19.    } else {
20.       d = x - **q2;
21.    }
22.  } else {
23.     d = d + 1;
24.  }
25.  z = x/d;
26.  print(a1+z)
29. }
```

(a) (2 points) List out the all the live definitions at line 20.
(4,a1)(5,a2)(6,p1)(7,p2)(8,q2)(10,a2)(12,a1)(14,q1)(16,x)(20,d)

(b) (1 point) Draw the *backward slice* starting with the use of **`d`** at line 25.



(c) (1 point) Is there any DVZ vulnerability in the above program? Explain using backward slice.
On line 25, DBZ possibly occurs. Backward slicing at line 25, d needs to be 0 for DBZ to occur. Line 20 possible sets d 0 if x is equal to **q2. X is set to be *p1(5) at line 16. Additionally, q2 points to a2 which is also 5. Therefore, there is a DBZ vulnerability in this program.

(d) (1 point) How many *def-use* (DU) chains starting with the definition of **`a2`** at line 5?

4 DU chains

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 O_1, O_2, O_3 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.

```
    X = malloc();
    Y = malloc();
    Z = malloc();
    If(c1) then q = &X
1       else if(c2) then q = &Y
2            else q = &Z
3   p = *q
```

At the Program Point 1:

T(X)={O_1} T(Y)={O_2} T(p)=null T(q)={O_2}

At the Program Point 2:

T(X)={O_1} T(Y)={O_2} T(p)=null T(q)={O_3}

At the Program Point 3:

T(X)={O_1} T(Y)={O_2} T(p)={O_3} T(q)={O_3}

```
    X = malloc();
    Y = malloc();
    p = malloc();
    If (a > 0) then q = &X
1       else q = &Y;
2   *q = p;
```

At the Program Point 1:

T(X)={O_1} T(Y)={O_2} T(p)={O_3} T(q)={O_2}

At the Program Point 2:

T(X)={O_1} T(Y)={O_2} T(p)={O_3} T(q)={p}

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

1. Suppose there are three non-nested branch 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?

   3

2. 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?

   3

Problem 4 (7 points): Access to the allocated memory is passed as a parameter `drptr` (a pointer to 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   Problem 2 Code Segment
2   dskenq(drptr, dsptr)
3       struct  dreq    *drptr;
4       struct  dsblk   *dsptr;
5   {
6       struct  dreq    *p, *q;
7       DBADDR  block;
8       int st;
9
10      if ( (q=dsptr->dreqlst) == DRNULL ) {
11          dsptr->dreqlst = drptr;
12          drptr->drnext = DRNULL;
13          dskstrt(dsptr);
14          return(DONQ);
15      }
16      block = drptr->drdba;
17      for (p = q->drnext ; p != DRNULL ; q=p,p=p->drnext) {
18          if (p->drdba==block && (st=dskqopt(p, q, drptr)!=SYSERR))
19                  return(st);
20          if ( (q->drdba <= block && block < p->drdba) ||
21               (q->drdba >= block && block > p->drdba)   ) {
22              drptr->drnext = p;
23              q->drnext = drptr;
24              return(DONQ);
25          }
26      }
27      drptr->drnext = DRNULL;
28      q->drnext = drptr;
29      return(DONQ);
30  }
```

A. **(2 points)** Does `dsptr` provide access to the allocated memory at line 13? If yes, explain in one sentence how.

Yes, it is because at line 11, dsptr->dreqlst is set equal to drptr.

B. (**1 point**) Does `dsptr` provide access to the allocated memory at line 16?

No

C. (**2 points**) How many definitions of `q` are there in the given code? Give the line numbers for those definitions.

2 definitions
Line 10, 17

D. (**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

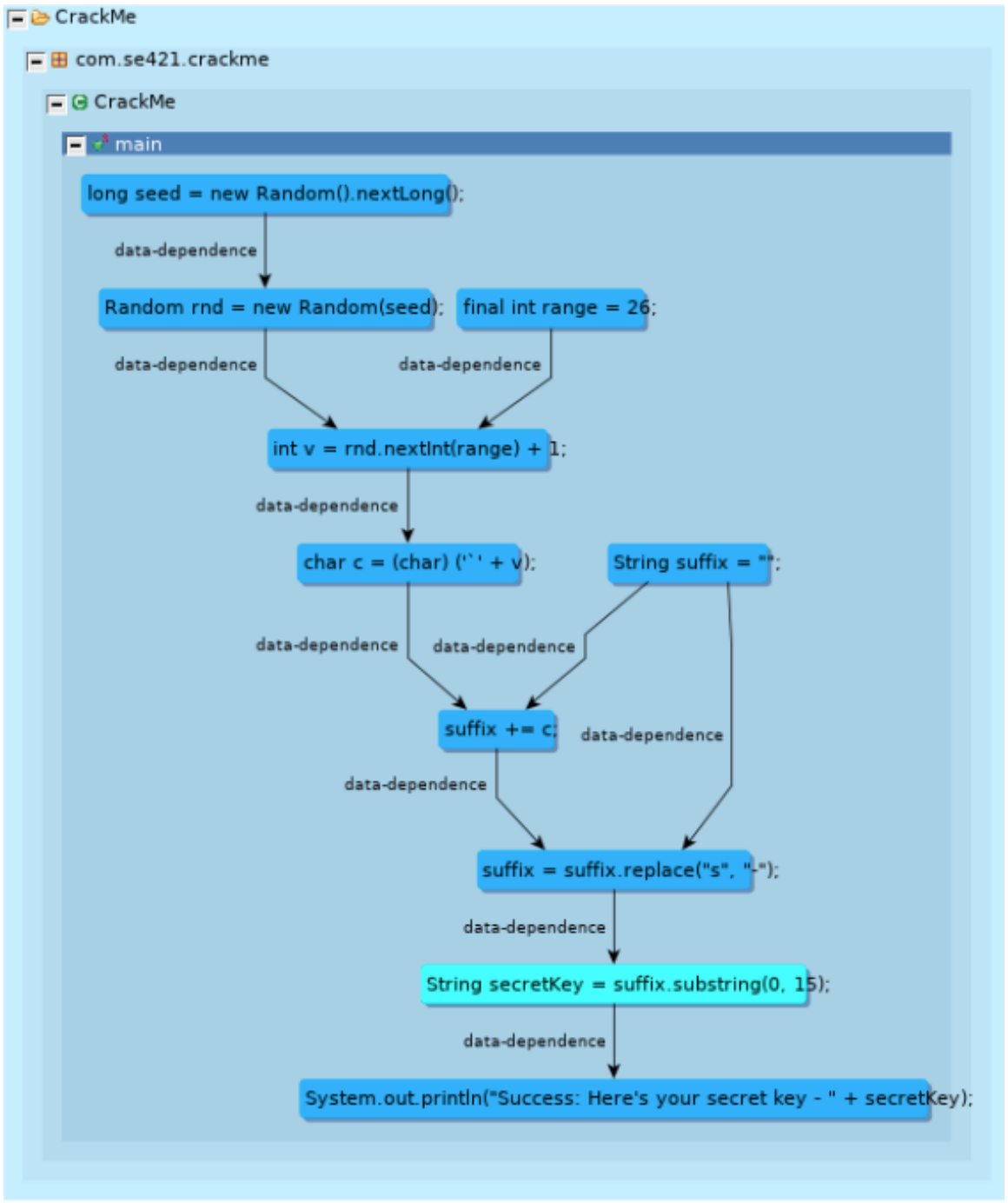
Problem 5 (2 points + 2 bonus points): *Import and map CrackMe project contained in crackme.zip*. The goal is to get the secret key generated by the given Java code. This problem was posed as a challenge problem at a cybersecurity competition. The problem is a difficult challenge if one tries fuzzing, which is a common practice at the cybersecurity competitions. Try running the code, you will find that it keeps running with periodic timer messages, but it does not produce the secret key.

The problem is not difficult to solve if you apply the program slicing technique we have studied in class. You can use the Program Slice Smart View (Data Dependence Slice as showed in class). Recall that you specified the appropriate program statement and the variable to compute the slice. An appropriate selection would yield the slice of the relevant code.

A. What are the appropriate statement and the variable for getting the program slice of the relevant code?

   Statement: String secretKey = suffix.substring(0,15);
   'suffix' is responsible for getting the relevant code that secret key is based on.

B. Include a graph of the program slice.


CrackMe
com.se421.crackme
CrackMe
main
long seed = new Random().nextLong();
data-dependence
Random rnd = new Random(seed);
final int range = 26;
data-dependence
data-dependence
int v = rnd.nextInt(range) + 1;
data-dependence
char c = (char) (`` ' + v);
String suffix = "";
data-dependence
data-dependence
suffix += c;
data-dependence
data-dependence
suffix = suffix.replace("s", "-");
data-dependence
String secretKey = suffix.substring(0, 15);
data-dependence
System.out.println("Success: Here's your secret key - " + secretKey);

C. (**Bonus Points 2**) Using the slice, remove the irrelevant code from the CrackMe.java and produce the secret key. Give your program and the secret key.

```java
package com.se421.crackme;

import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Random;

public class CrackMe {

    public static void main(String[] args) throws Exception {
        while (true) {
            long seed = new Random().nextLong();
            Random rnd = new Random(seed);
            final int range = 26;
            String suffix = "";
            for (int i = 0; i < 20; i++) {
                int v = rnd.nextInt(range) + 1;
                char c = (char) ('`' + v);
                suffix += c;
            }
            suffix = suffix.replace("s", "-");
            if (SHA256(suffix).startsWith("e9")) {
                String secretKey = suffix.substring(0, 15);
                System.out.println("Success: Here's your secret key - " + secretKey);
                break;
            }
        }
    }

    private static String SHA256(String input) throws NoSuchAlgorithmException {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(input.getBytes(StandardCharsets.UTF_8));
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < hash.length; i++) {
            String hex = Integer.toHexString(0xff & hash[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }

}
```

```
Success: Here's your secret key - bmvmpblrweczyxw
```