

Assignment 1 - CT255 CyberSecurity

Daniel Hannon (19484286)

November 2020

1 Problem 2

```
1  /**
2   *
3   * @author Michael Schukat, 2019 modified by Daniel Hannon
4   */
5  import java.util.Date;
6
7  public class CT255_HashFunction1 {
8
9      public static void main(String[] args) {
10         int res = 0;
11         int res1 = 0;
12         int collisions_found = 0;
13
14         if (args != null && args.length > 0) { // Check for <input> value
15             res = hashF1(args[0]); // call hash function with <input>
16             if (res < 0) { // Error
17                 System.out.println("Error: <input> must be 1 to 64 characters long.");
18             }
19             else {
20                 System.out.println("input = " + args[0] + " : Hash = " + res);
21                 Date time = new Date();
22                 long timeStart = time.getTime();
23                 System.out.println("Start searching for collisions");
24                 while(collisions_found < 10) {
25                     String test = "";
26                     for(int i = 0; i < 5; i++) { /*Length 5 string*/
27                         test+=Character.toString((char)( Math.random() * 78)+48);
28                         // /*ASCII value "0" is 48
29                         // and "~" is 126*/
30                     }
31                     res1 = hashF1(test);
32                     if(res == res1 && args[0].equals(test) == false) {
33                         // /*Compares hashes and strings to
34                         // make sure that it is not the exact same string if there's a duplicate*/
35                         time = new Date();
36                         long currtime = time.getTime() - timeStart;
37                         System.out.println(currtime+"ms: Collision Found! " + test);
38                         collisions_found++;
39                     }
40                 }
41             }
42         }
43         else { // No <input>
44             System.out.println("Use: CT255_HashFunction1 <Input>");
45         }
46     }
47
48     private static int hashF1(String s){
49         int ret = -1, i;
50         int[] hashA = new int[]{1, 1, 1, 1};
51
52         String filler, sIn;
```

```

51
52     filler = new
53         ↪ String("ABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGH");
54
55     if ((s.length() > 64) || (s.length() < 1)) { // String does not have required length
56         ret = -1;
57     }
58     else {
59         sIn = s + filler; // Add characters, now have "<input>ABCDEFGH..."
60         sIn = sIn.substring(0, 64); // Limit string to first 64 characters
61         // System.out.println(sIn); // FYI
62         for (i = 0; i < sIn.length(); i++){
63             char byPos = sIn.charAt(i); // get i'th character
64             hashA[0] += (byPos * 17); // Note: A += B means A = A + B
65             hashA[1] += (byPos * 31);
66             hashA[2] += (byPos * 101);
67             hashA[3] += (byPos * 79);
68         }
69
70         hashA[0] %= 255; // % is the modulus operation, i.e. division with rest
71         hashA[1] %= 255;
72         hashA[2] %= 255;
73         hashA[3] %= 255;
74
75         ret = hashA[0] + (hashA[1] * 256) + (hashA[2] * 256 * 256) + (hashA[3] * 256 * 256 *
76             ↪ 256);
77         if (ret < 0) ret *= -1;
78     }
79     return ret;
80 }
81 }

```

2 Problem 3

```
/**
 *
 * @author Michael Schukat, 2019 Modified by Daniel Hannon
 */
import java.util.Date;

public class CT255_HashFunction2 {

    public static void main(String[] args) {
        int res = 0;
        int res1 = 0;
        int collisions_found = 0;

        if (args != null && args.length > 0) { // Check for <input> value
            res = hashF2(args[0]); // call hash function with <input>
            if (res < 0) { // Error
                System.out.println("Error: <input> must be 1 to 64 characters long.");
            }
            else {
                System.out.println("input = " + args[0] + " : Hash = " + res);
                /*Added time it takes to compare Algorithms*/
                Date time = new Date();
                long timeStart = time.getTime();
                System.out.println("Start searching for collisions");
                while(collisions_found < 10) {
                    String test = "";
                    for(int i = 0; i < 5; i++) { /*Length 5 string*/
                        test+=Character.toString((char)(Math.random() * 78)+48);
                        ↪ /*ASCII value "0" is 48
                           and "~" is 126*/
                    }
                    res1 = hashF2(test);
                    if(res == res1 && args[0].equals(test) == false) {
                        ↪ /*Compares hashes and strings to
                           make sure that it is not the exact same string if there's a duplicate*/
                        time = new Date();
                        /*Get Time of collision relative to program start*/
                        long timecurr = time.getTime() - timeStart;
                        System.out.println(timecurr+"ms: Collision Found! " + test);
                        collisions_found++;
                    }
                }
            }
        }
        else { // No <input>
            System.out.println("Use: CT255_HashFunction1 <Input>");
        }
    }

    private static int hashF2(String s){
        int ret = -1, i;
        int[] hashA = new int[]{1, 1, 1, 1};

        String filler, sIn;
        char space1,space2,space3;
        /* By swapping around one AB in this I have completely removed several duplicat
        filler = new
        ↪ String("ABCDEFGBACDEFGHABCFEFGHABCFEFGHABCFEFGHABCFEFGHABCFEFGHABCFEFGH")
```

```

56 char[] primes = {17,79,91,103,83,67,101,47,89,107,19,41};
   ↪ /*Random primes in a random order*/
57 char[] primes2 = {91,19,107,101,83,89,41,17,103,79,67,47};
   ↪ /*Different Order of the same primes*/
58
59 if ((s.length() > 64) || (s.length() < 1)) { // String does not have required length
60     ret = -1;
61 }
62 else {
63     sIn = s + filler; // Add characters, now have "<input>HABCDEF..."
64     sIn = sIn.substring(0, 64); // Limit string to first 64 characters
65     space1 = sIn.charAt(5);
66     space2 = sIn.charAt(11);
67     space3 = sIn.charAt(43);
68
69     // System.out.println(sIn); // FYI
70     for (i = 0; i < sIn.length(); i++){
71         char byPos = sIn.charAt(i); // get i'th character
72         /*This works like a somewhat Primitive Enigma Device wherein the number
73         that the value is changed by depends heavily on the position of the letter
74         within the string, to introduce some extra challenge to crack it
75         the displacement of the 2nd through 4th rotors depends on the values
76         held at fixed arbitrary positions within the string, not only this, the length
77         of the rotor which is operated is staggered.
78         To reduce the amount of collisions I also changed the filler text to include
79         "BACDEFGFH" in one location so duplicates like STRINGABCDEFGFH would no longer
80         exist.*/
81         hashA[0] += (byPos * (primes[i % 12]));
82         hashA[1] += (byPos * (primes[(i + space1) % 11]));
83         hashA[2] += (byPos * (primes2[(i + space2) % 12]));
84         hashA[3] += (byPos * (primes2[(i + space3) % 11]));
85     }
86     hashA[0] %= 255; // % is the modulus operation, i.e. division with rest
87     hashA[1] %= 255;
88     hashA[2] %= 255;
89     hashA[3] %= 255;
90
91     ret = hashA[0] + (hashA[1] * 256) + (hashA[2] * 256 * 256) + (hashA[3] * 256 * 256 *
   ↪ 256);
92     if (ret < 0) ret *= -1;
93 }
94 return ret;
95 }
96 }

```