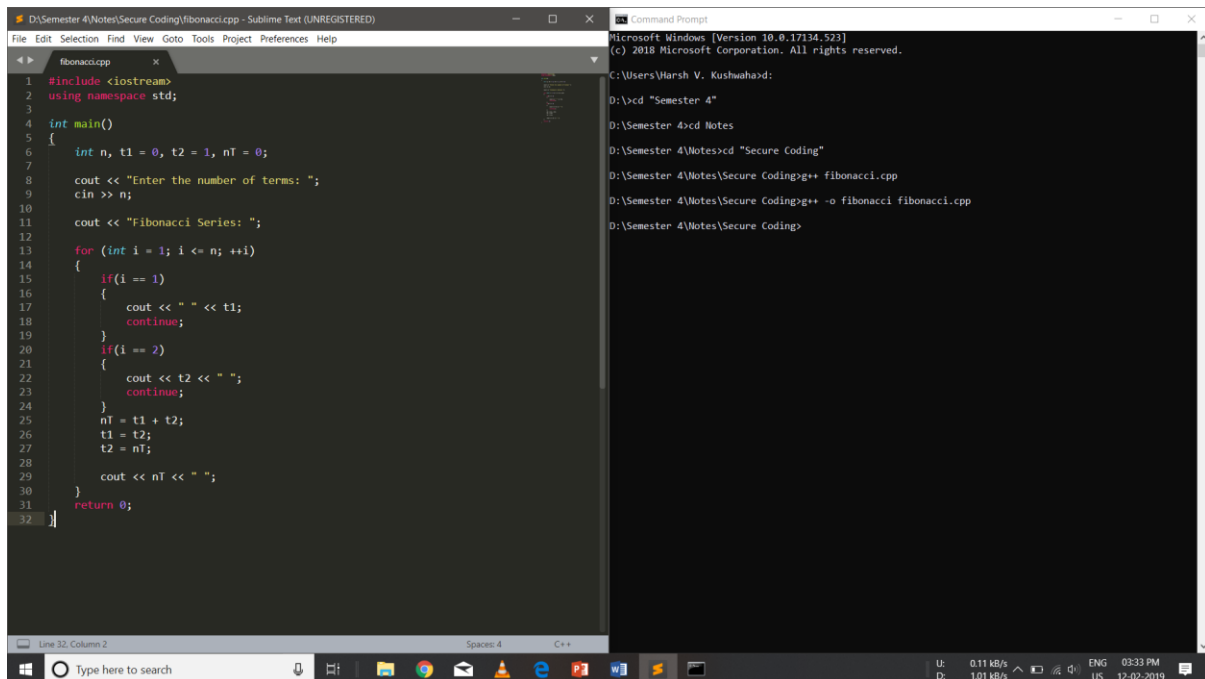


Harsh Vardhan Kushwaha

17BCN7017

Class Work – 1

1. Write a CPP code, compile and create an executable.



The screenshot displays a Windows desktop environment. On the left, a Sublime Text editor window titled 'D:\Semester 4\Notes\Secure Coding\fibonacci.cpp - Sublime Text (UNREGISTERED)' contains the following C++ code:

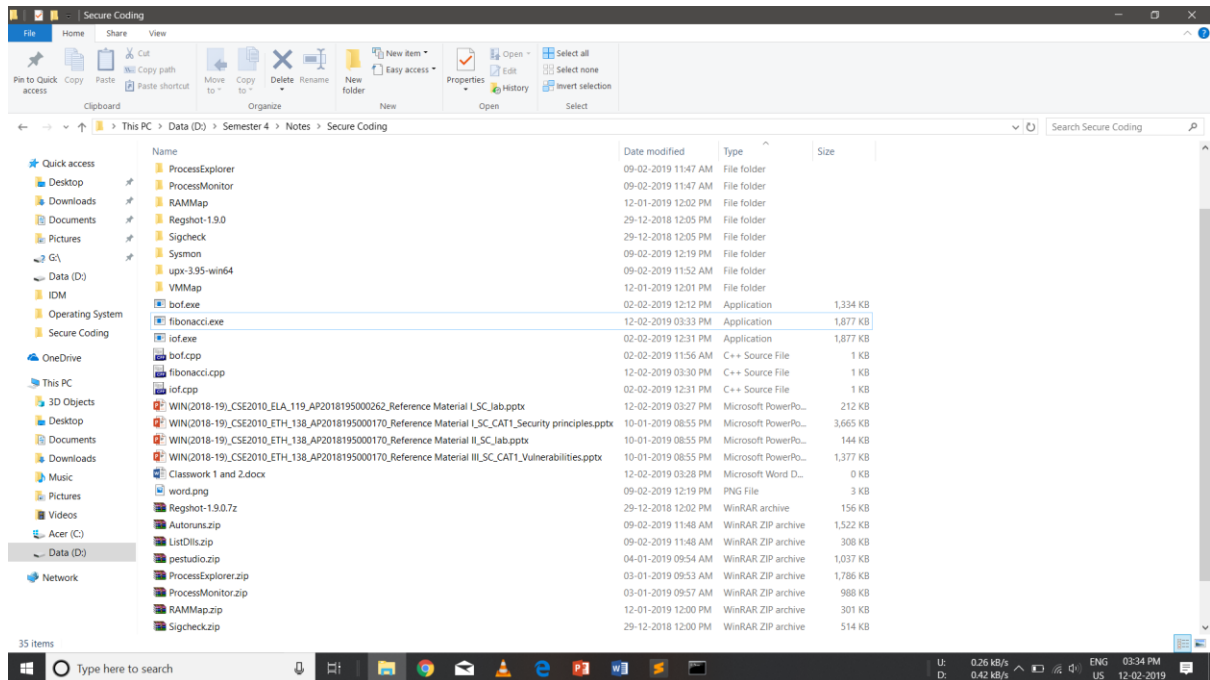
```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int n, t1 = 0, t2 = 1, nT = 0;
7
8     cout << "Enter the number of terms: ";
9     cin >> n;
10
11     cout << "Fibonacci Series: ";
12
13     for (int i = 1; i <= n; ++i)
14     {
15         if(i == 1)
16         {
17             cout << " " << t1;
18             continue;
19         }
20         if(i == 2)
21         {
22             cout << t2 << " ";
23             continue;
24         }
25         nT = t1 + t2;
26         t1 = t2;
27         t2 = nT;
28
29         cout << nT << " ";
30     }
31     return 0;
32 }
```

On the right, a Command Prompt window titled 'Microsoft Windows [Version 10.0.17134.523] (c) 2018 Microsoft Corporation. All rights reserved.' shows the following commands and their output:

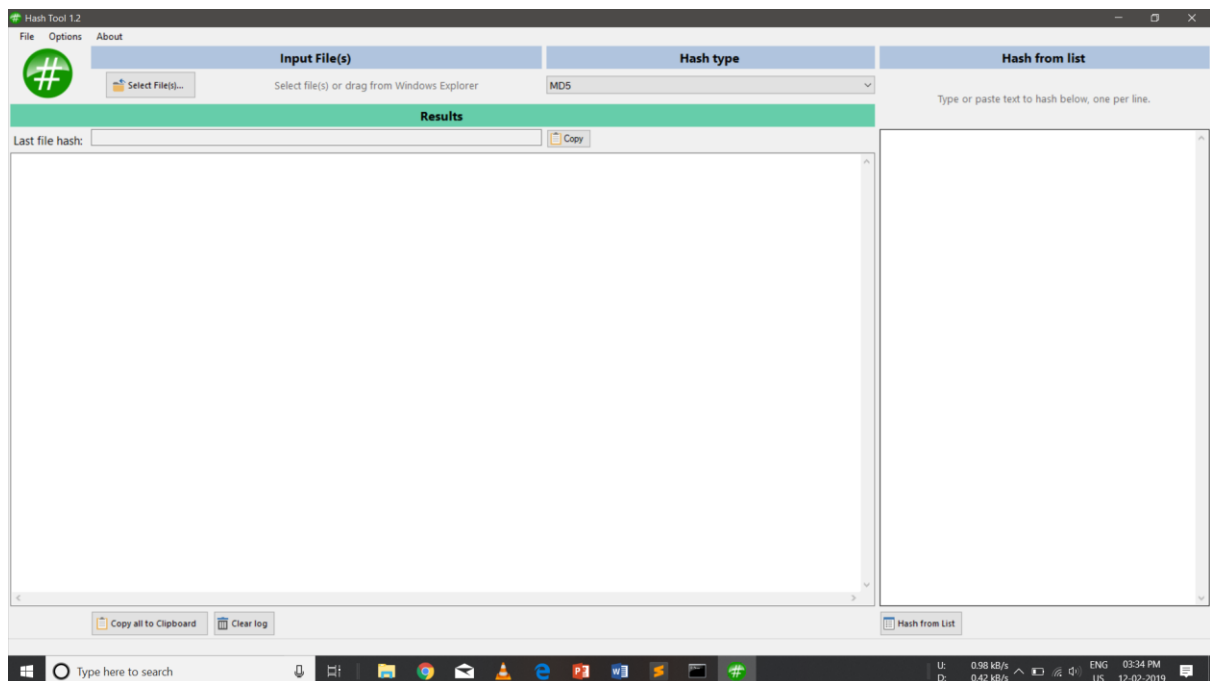
```
C:\Users\Harsh V. Kushwaha>
D:\>cd "Semester 4"
D:\Semester 4>cd Notes
D:\Semester 4\Notes>cd "Secure Coding"
D:\Semester 4\Notes\Secure Coding>g++ fibonacci.cpp
D:\Semester 4\Notes\Secure Coding>g++ -o fibonacci fibonacci.cpp
D:\Semester 4\Notes\Secure Coding>
```

The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system tray on the right indicates the date and time as 12-02-2019, 03:33 PM.

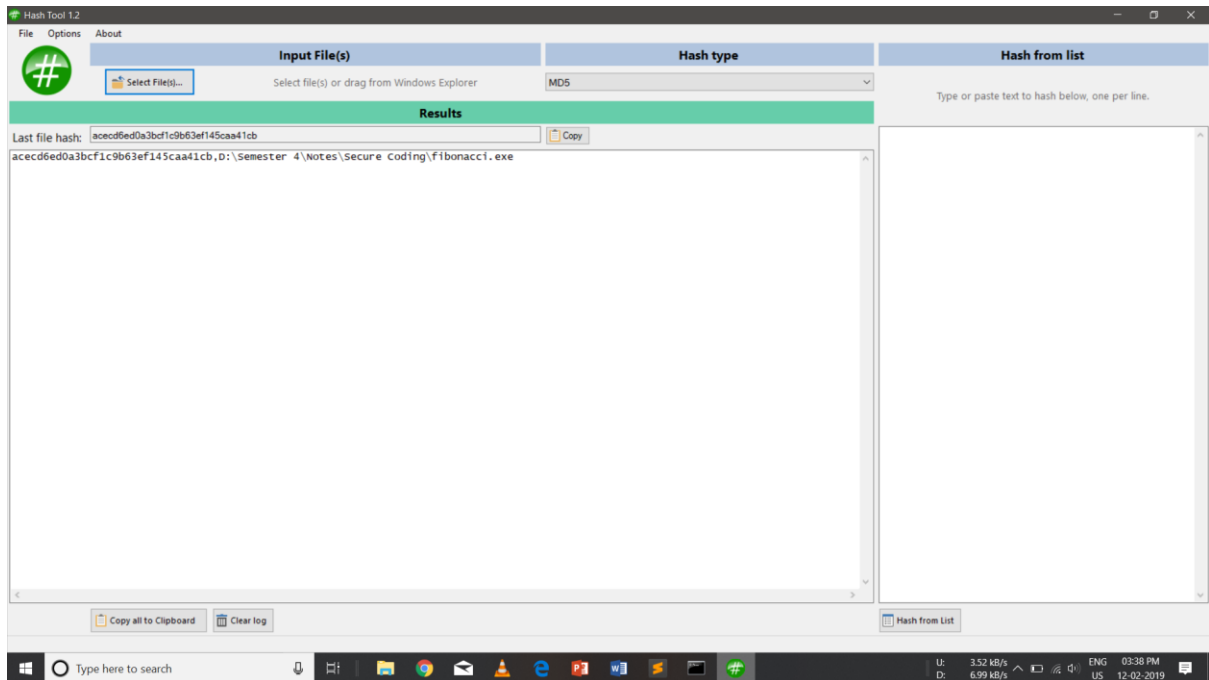
2. Created Exe



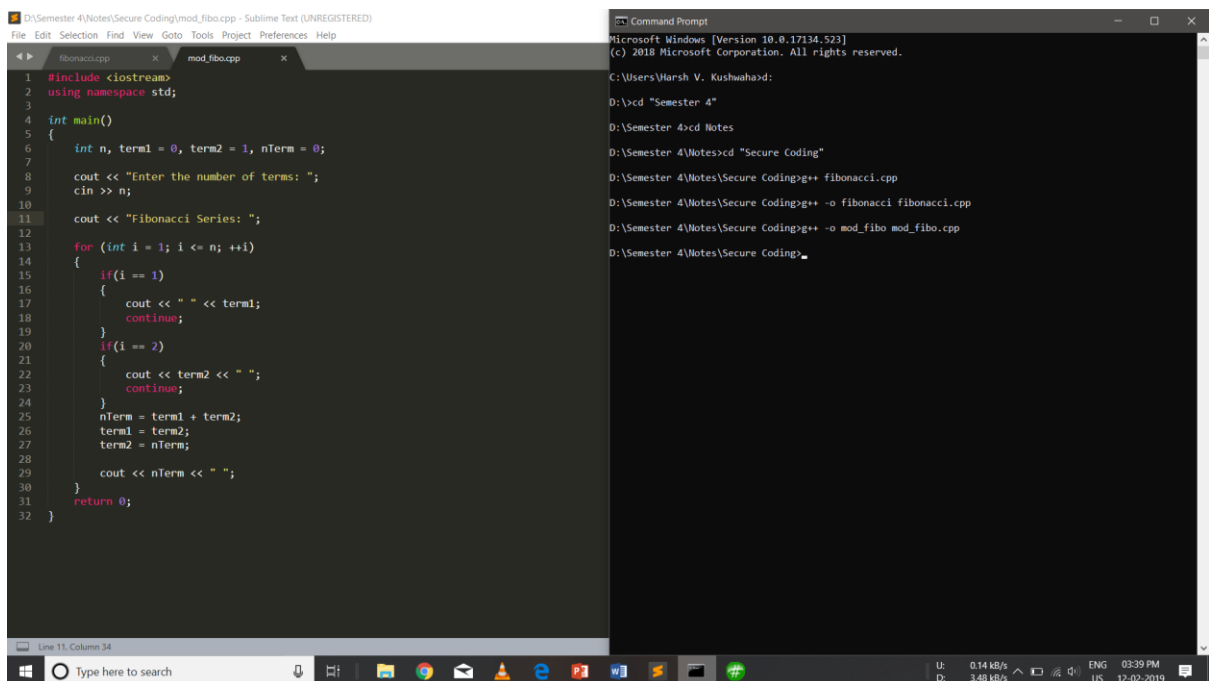
3. Download any Hash tool. (I used the one provided in Windows 10 App Store)



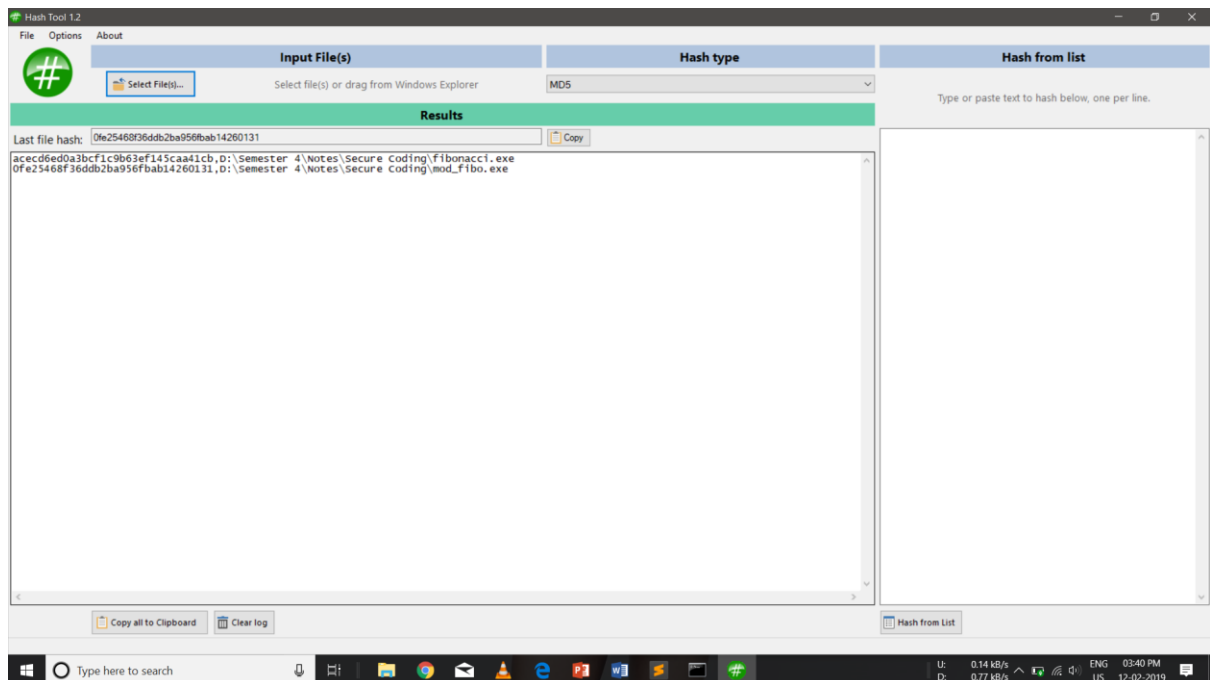
4. Generate the signature for the executable.



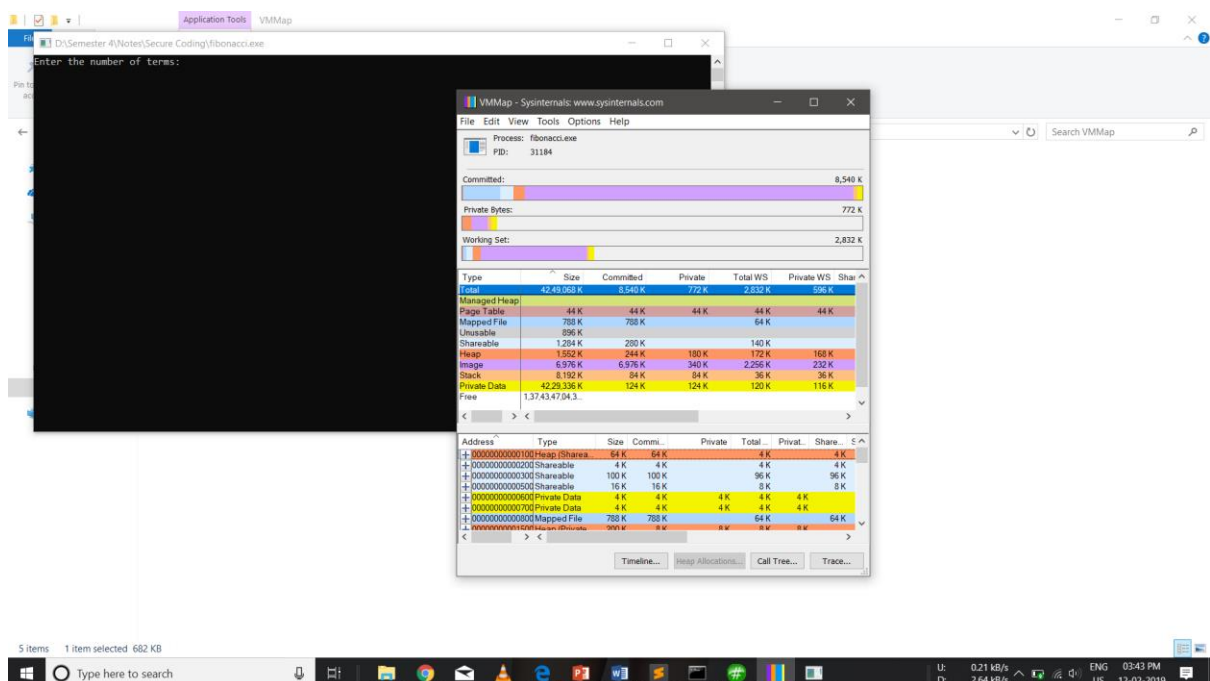
5. Modify the existing code. I changed the variables name.



6. Generate the signature for modified file and compare the hash strings.



7. Inspecting the memory using the sysinternal tool VMMap.



8. Inspecting the physical memory and processes using RAMMap.

