Brian Chase
(and Matt, Zach, Jonathan, and Rebecca!)
CPS 592-N1 Spring 2017
Lab 6 Report

Problem 1:

Originally, after tracking the calls through IDA Pro, we thought we should use SetWindowsHookEx() to intercept the message box and change the string that it had. After several unsuccessful attempts to implement SetWindowsHookEx() and a couple emails to Professor Deep, we decided to use VirtualProtect() to overwrite the string at its source. After tracking the string through PE Studio, hex editor, and Resource Hacker, we calculated the raw offset. Once we were able to locate the original message, we could rewrite it with one of our own.

Problem 2:

Looked at IDA PRo and set breakpoints and discovered what values were being changed with the SetInt() function. Within the SetInt() function was a call to RegSetValueExW() Windows API function. In the SetInt function where it calls to RegCreateKeyW and RegCloseKeyW(), in order to change the number of wins, we needed the registry key number and the path to the registry key, which were pushed before calls to RegCreateKey(). _pszWon value was the registry key value that stored the number of wins.

Problem 3:

First we discovered that the MainWndProc function in the IDA Pro disassembly clearly represented a switch statement. After a little investigating, we discovered that this switch statement handled all the keyboard short-cuts. We set up breakpoints to find where Ctrl-Shift_F10 would pop-up. When we finally located the Ctrl-Shift_F10 case, we were able to locate the prompt that comes up and choose "Abort, Retry, Cancel". We followed the path taken by the Abort choice and found a function called cheating. Following the path of this function led us to discover that setting Cheating to 2 forces a win. In the code, we set this value to 2.

Problem 4:

We explored a couple different solutions to this problem. One solution was to implement a kind of listener using a loop that would monitor for a certain keyboard combinations. To set the combinations of the new keyboard short-cuts, we used the GetKeyState() function. By using this function, we could override the accelerator table that the Freecell program uses.

Another solution was to use SetWindowsHookEx(). The idea was to use this to intercept the keyboard commands and to return a result of our own choosing. However, like with Problem 1, we were unable to figure out how to implement it.

Brian Chase
(and Matt, Zach, Jonathan, and Rebecca!)
CPS 592-N1 Spring 2017
Lab 6 Report

Problem 5:

Because of our difficulties with Problem 4, we were unable to solve problem 5 with the accelerator table. Since we couldn't get the accelerator table working, we went the action listener instead. The down side of doing it this way is that the dll file will keep running after the free cell executable is closed. We used GetKeyState for each of the keys required and logically anded them together to check if the keys were down. If they were we called, MoveCards from Freecell to force a win.