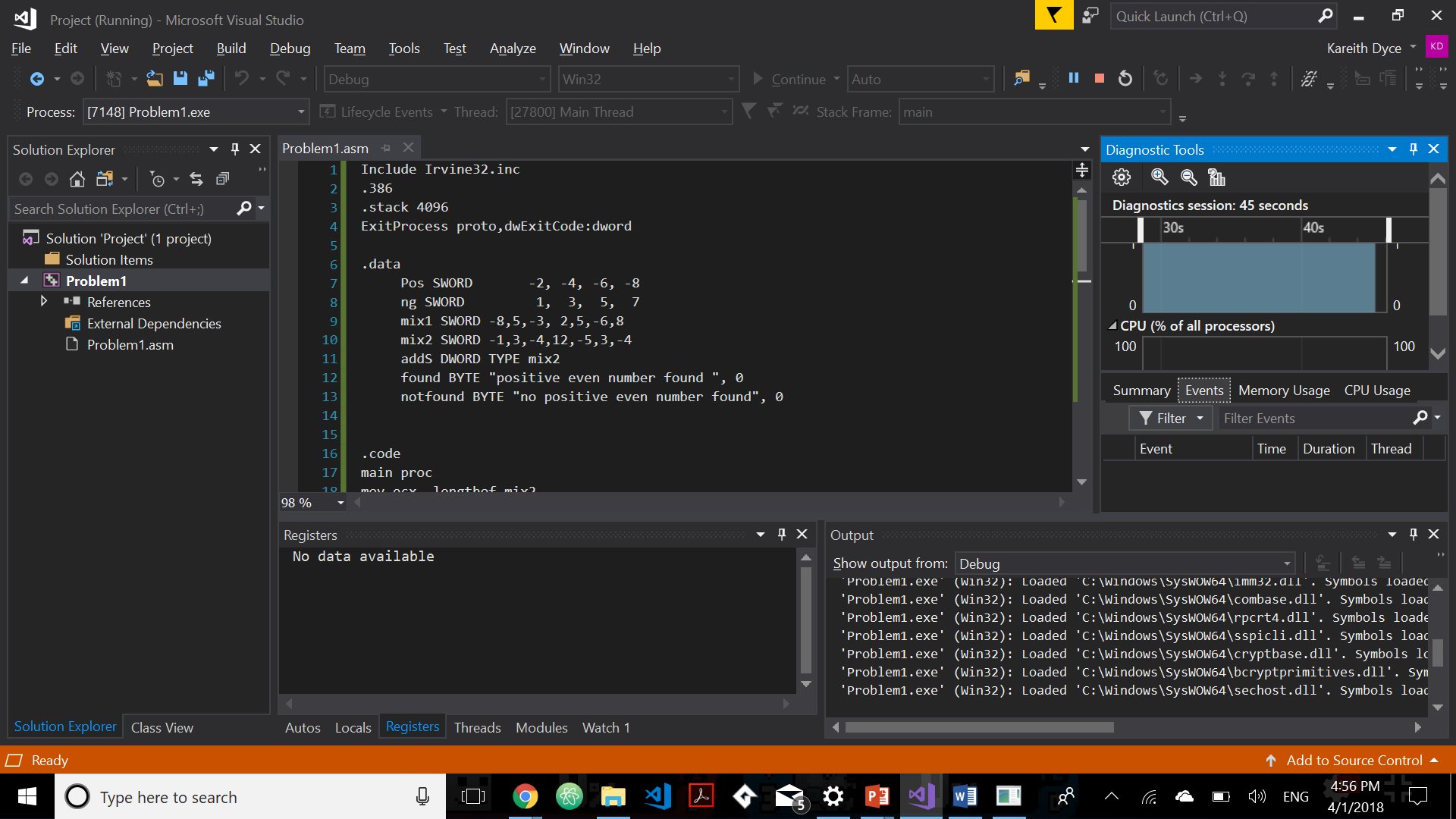
# Comp 3350: Computer Organization & Assembly Language

# HW # 7: Theme: Conditionals, Booleans, Loops

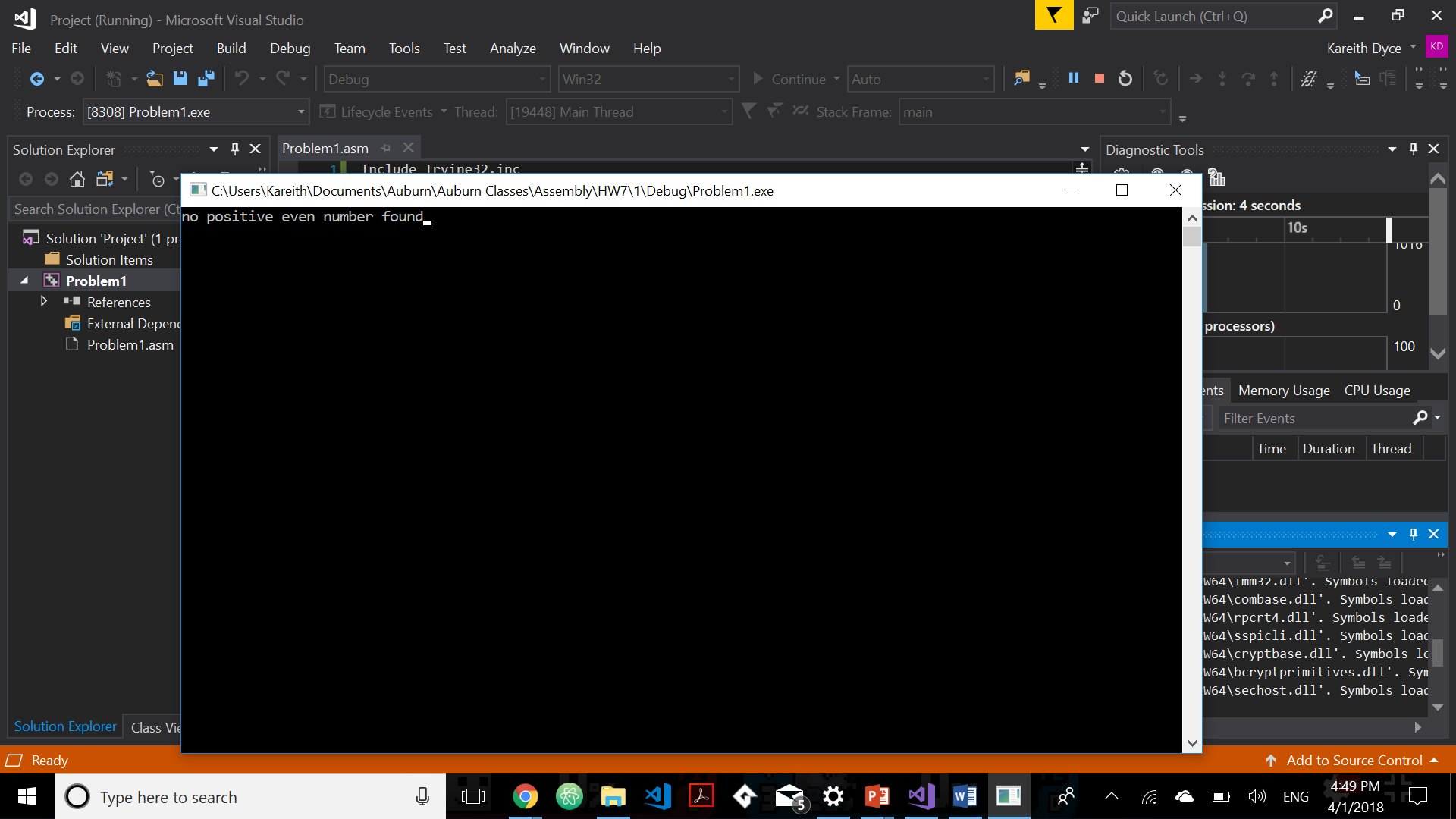
*(All main questions carry equal weight. Credit awarded to only those answers for which work has been shown.)*

1. Draft a program that scans an array to determine the first positive EVEN number in the array. If a positive value is found, the program should print “positive even number found” and the value. If no positive EVEN value is found in the array, the program should print “no positive even number found.” Submit list file and show the runs for the following data items:
   1. all negative even values
   2. all positive odd values
   3. mixed negative and positive values which are odd and even (two different examples with odd and even numbers at different indices)

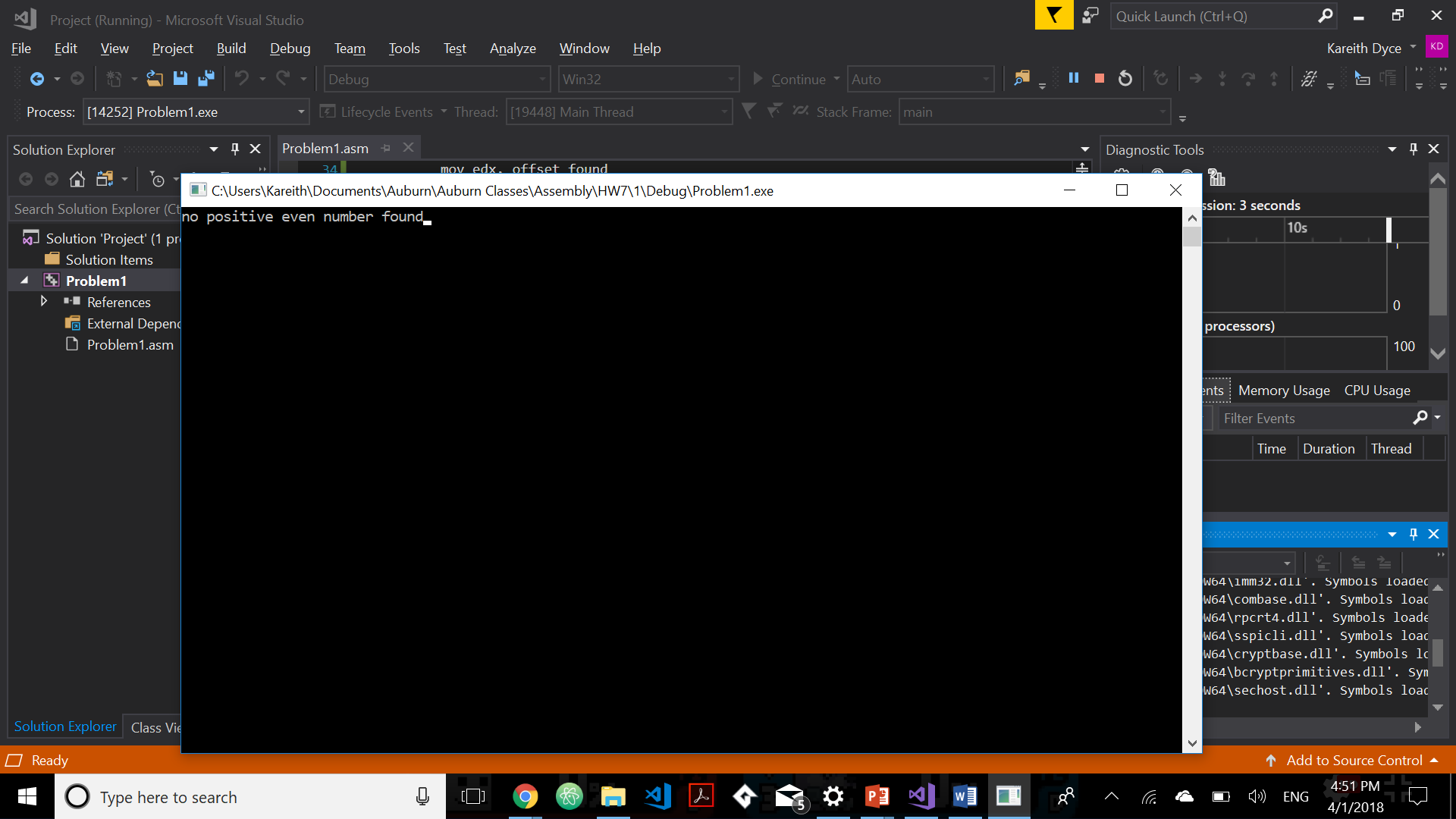
The arrays used



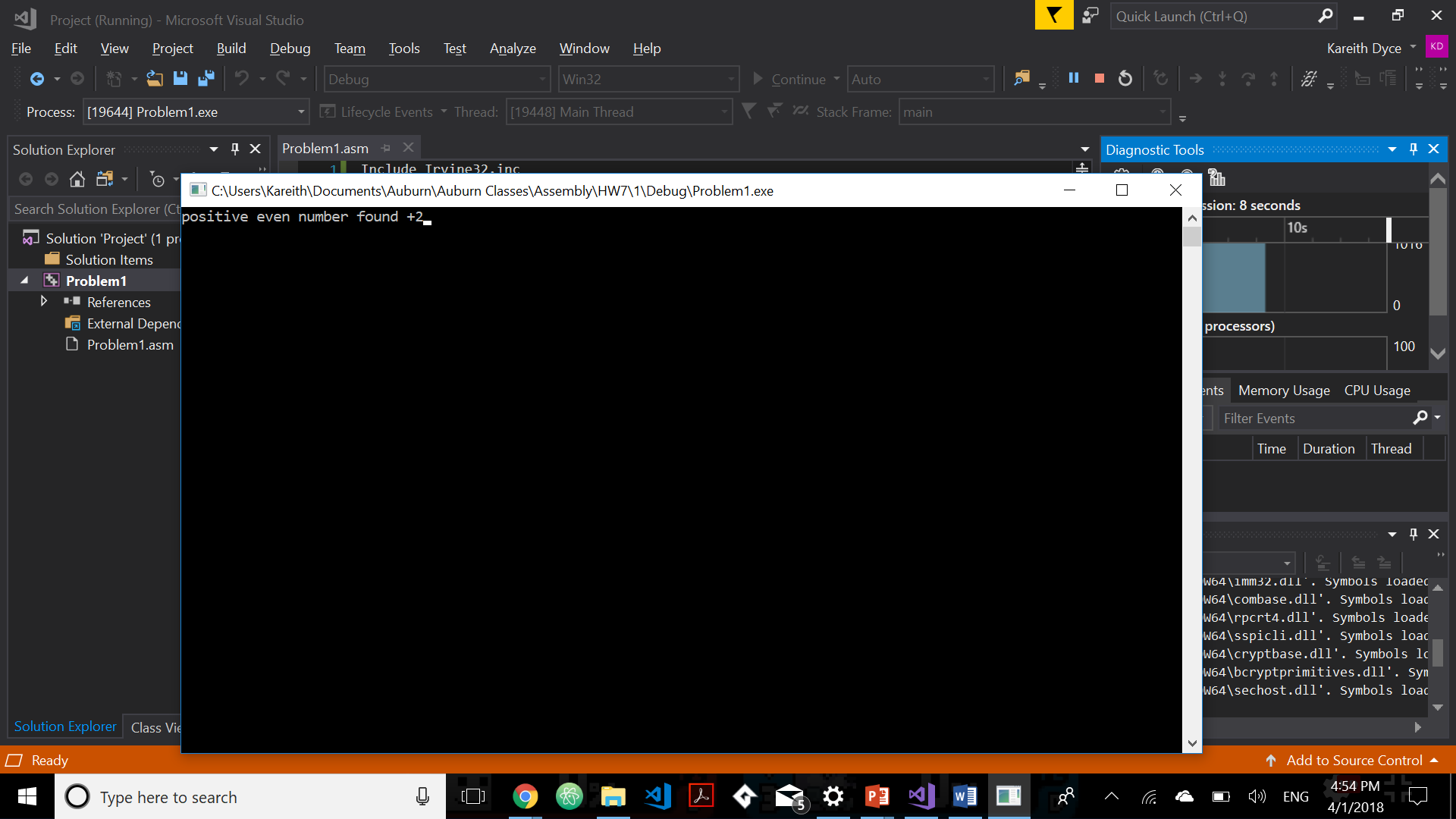
Array 1: (pos) Results



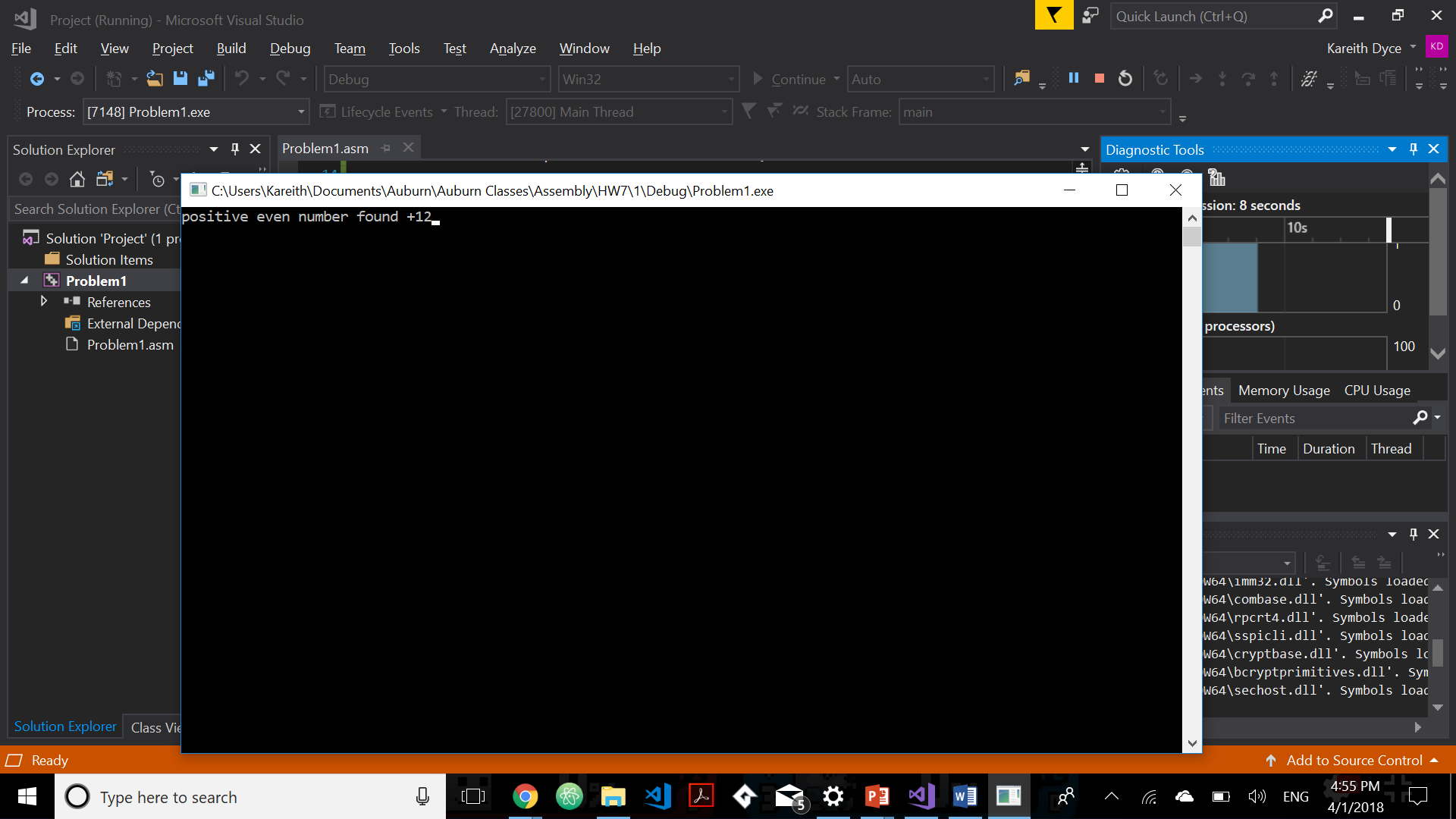
Array 2: (ng) Results

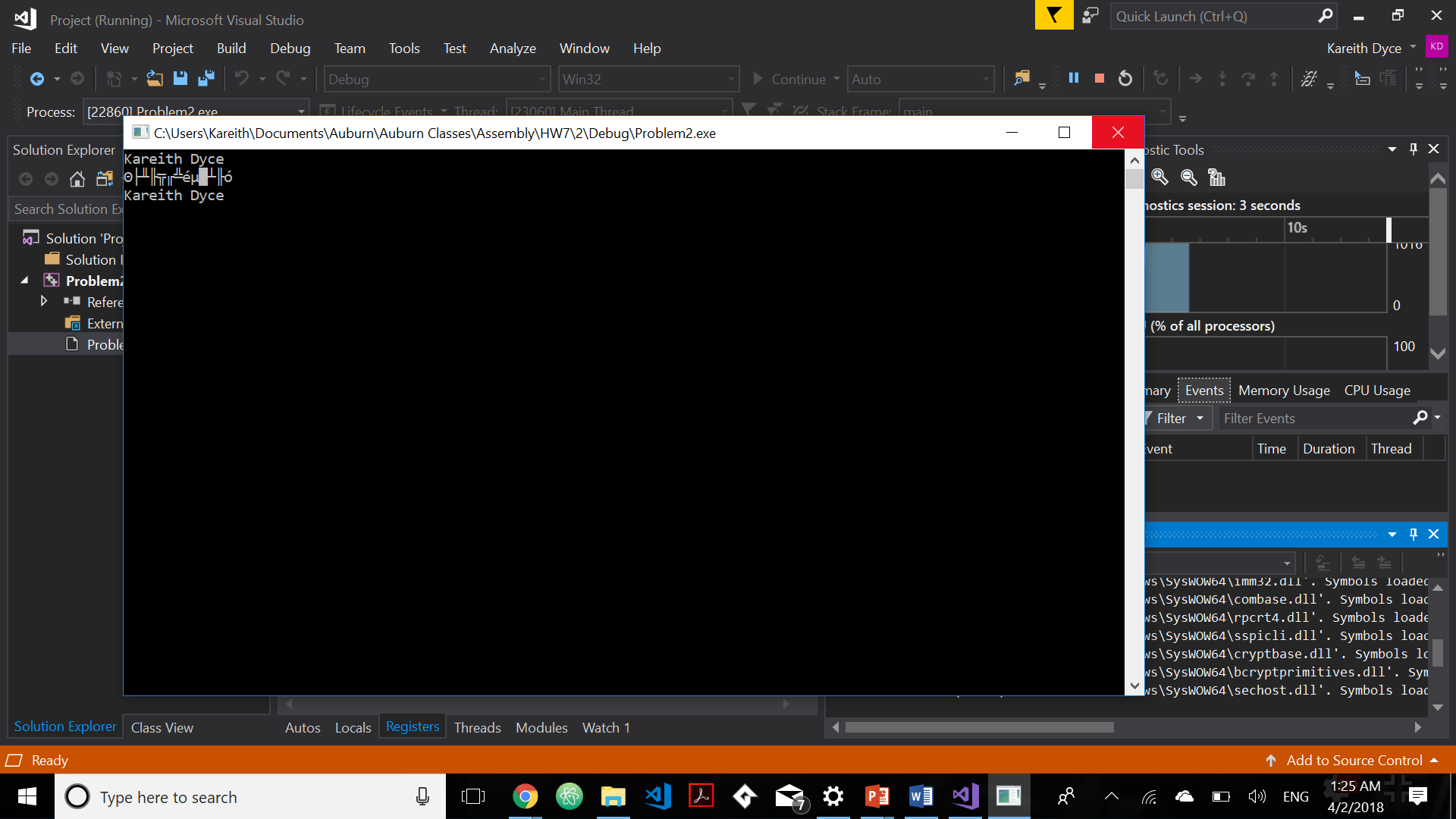


Array 3: (mix1) Results



Array 4: (mix2) Results



1. Write a program which encodes any string using the XOR instruction. Test it using your <first name last name> in the data segment to produce cipher text and then decode using the program to get plain text. Use the last three digits of your student id as the key. Print plane text from the data segment, print the cipher text, and then print the plain text upon execution. What are the strengths and weaknesses of this encryption method? Can you think of another way doing such encryption? What are the strengths and weaknesses of your method? The strength is that this method is easy and fast to enact. The weakness of this method is that since it is so simple it can be easily cracked. A better way would be to double the key after each letter. This would make it slightly harder to decrypt. However it would be harder to enact and take to longer to encrypt. 

1. Implement the following two pseudo-codes in assembly language (assume signed numbers). Declare Apple and Pear as word sized variables. Test the program for input data sets listed below and print values assigned to Apple and Pear. Submit list file and show output for all input data sets.
2. if ( (cx = bx) AND (cx >= val1) )

Apple = 1;

Else

Apple = 0;

1. if ( (bx = cx) OR (cx >= val1) )

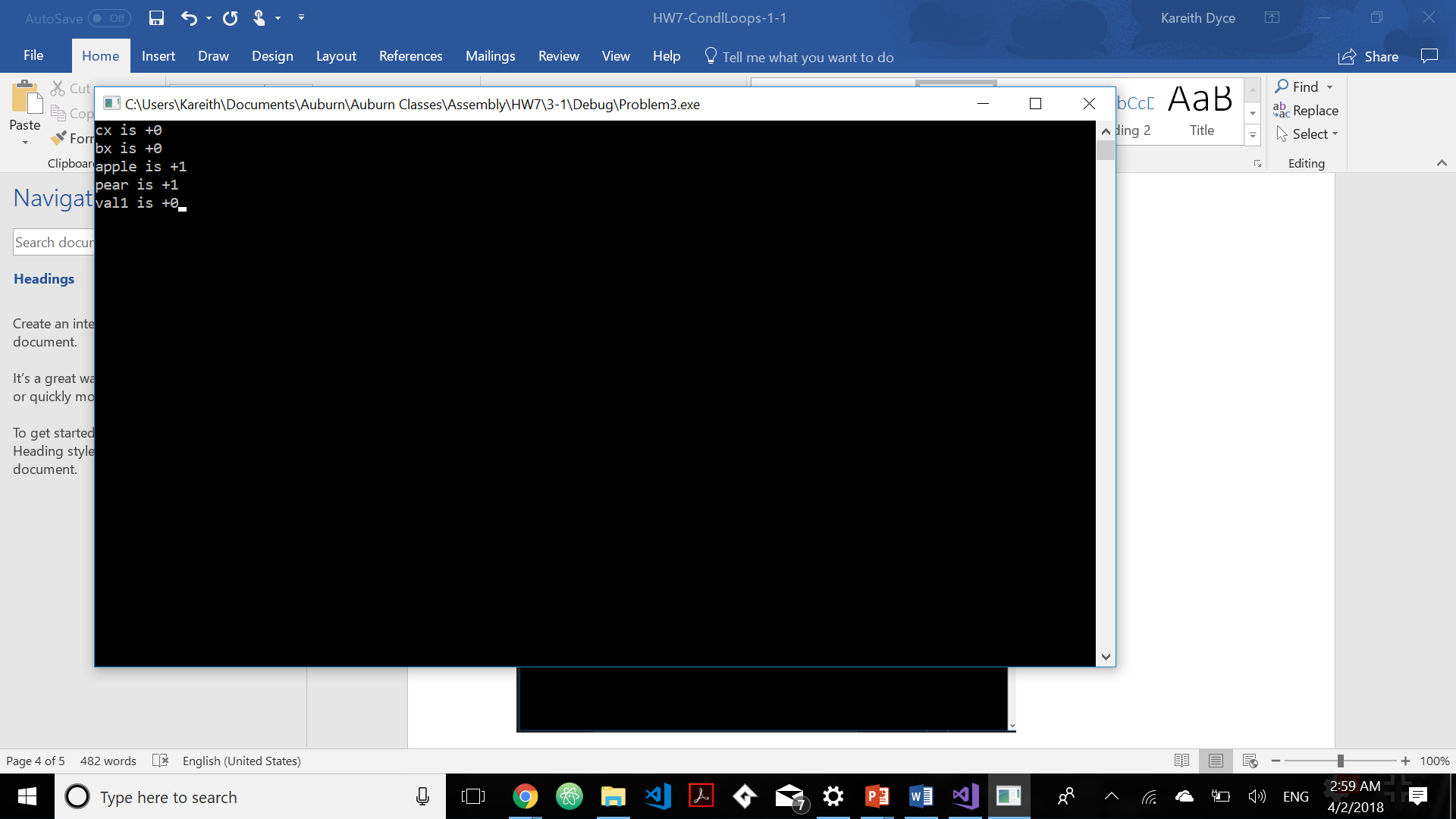
Pear = 1;

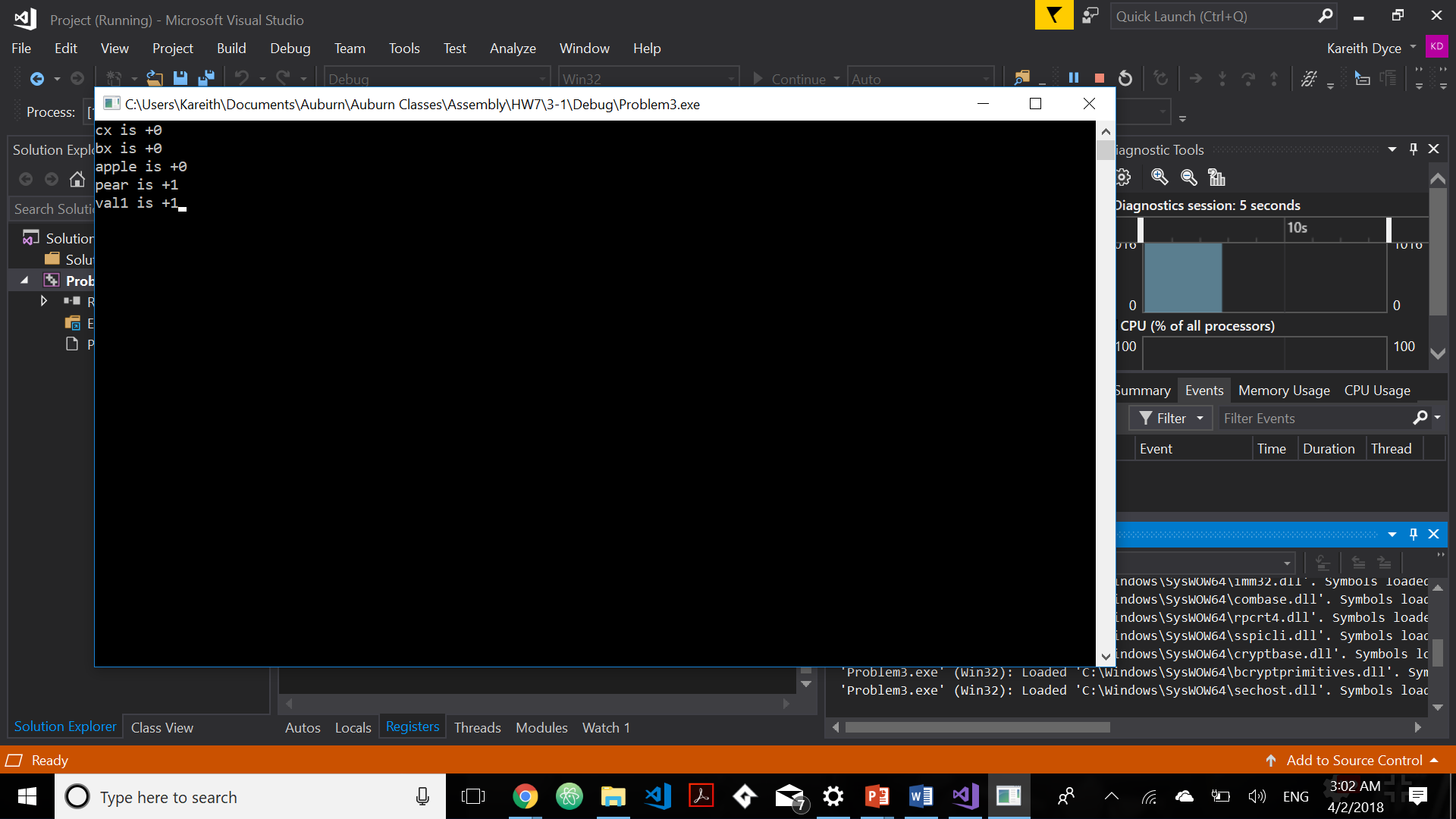
Else

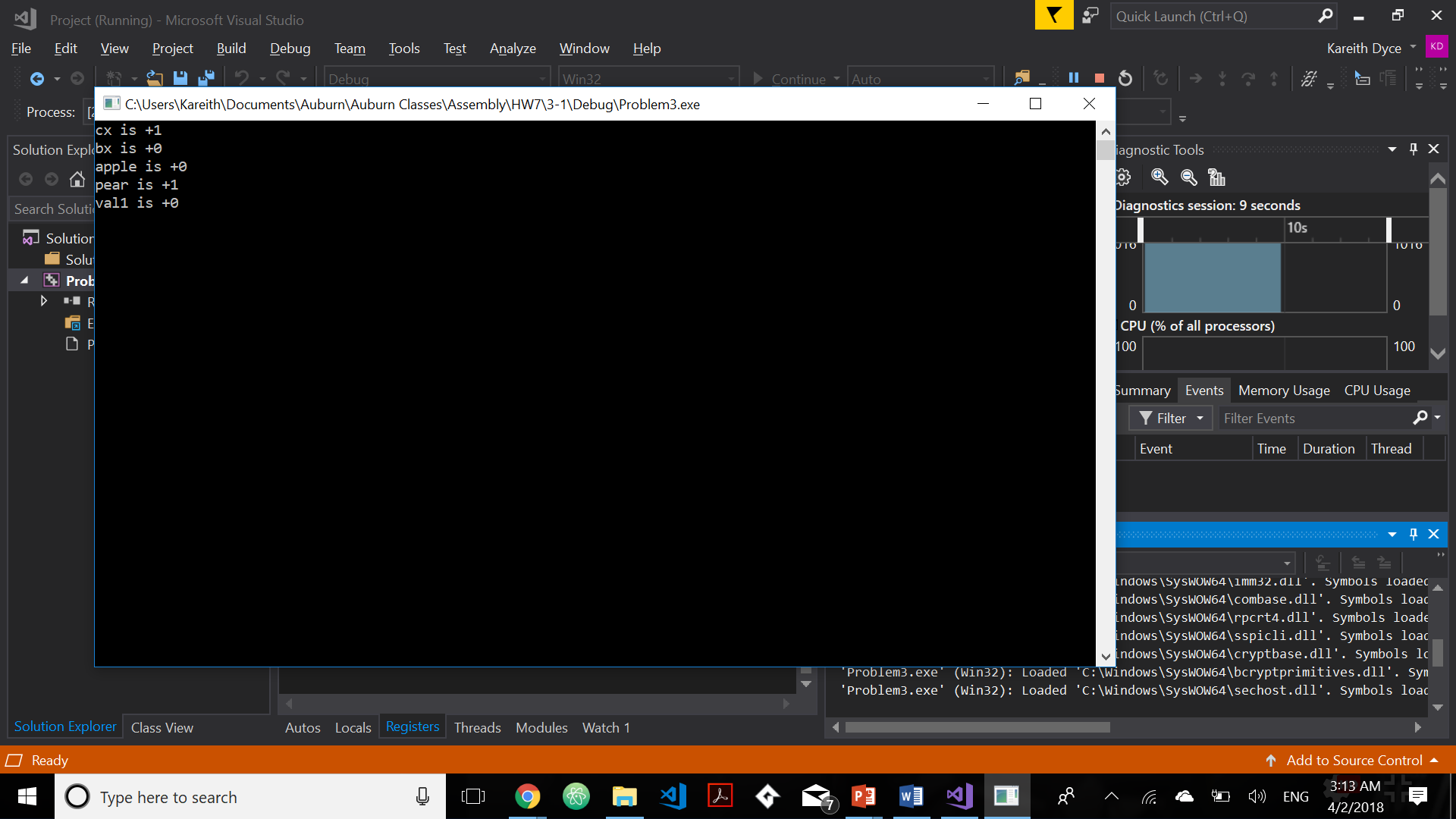
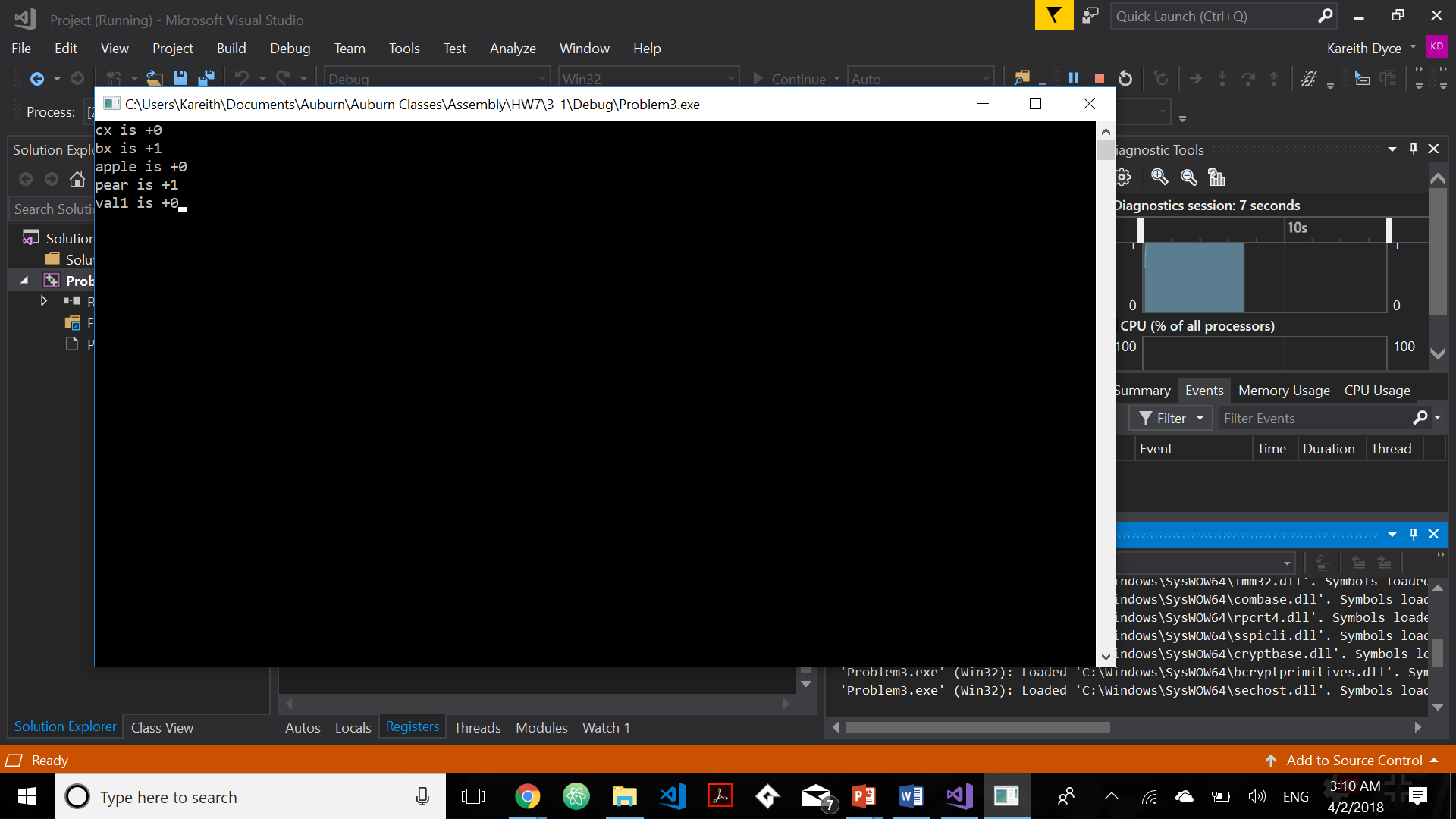
Pear = 0;

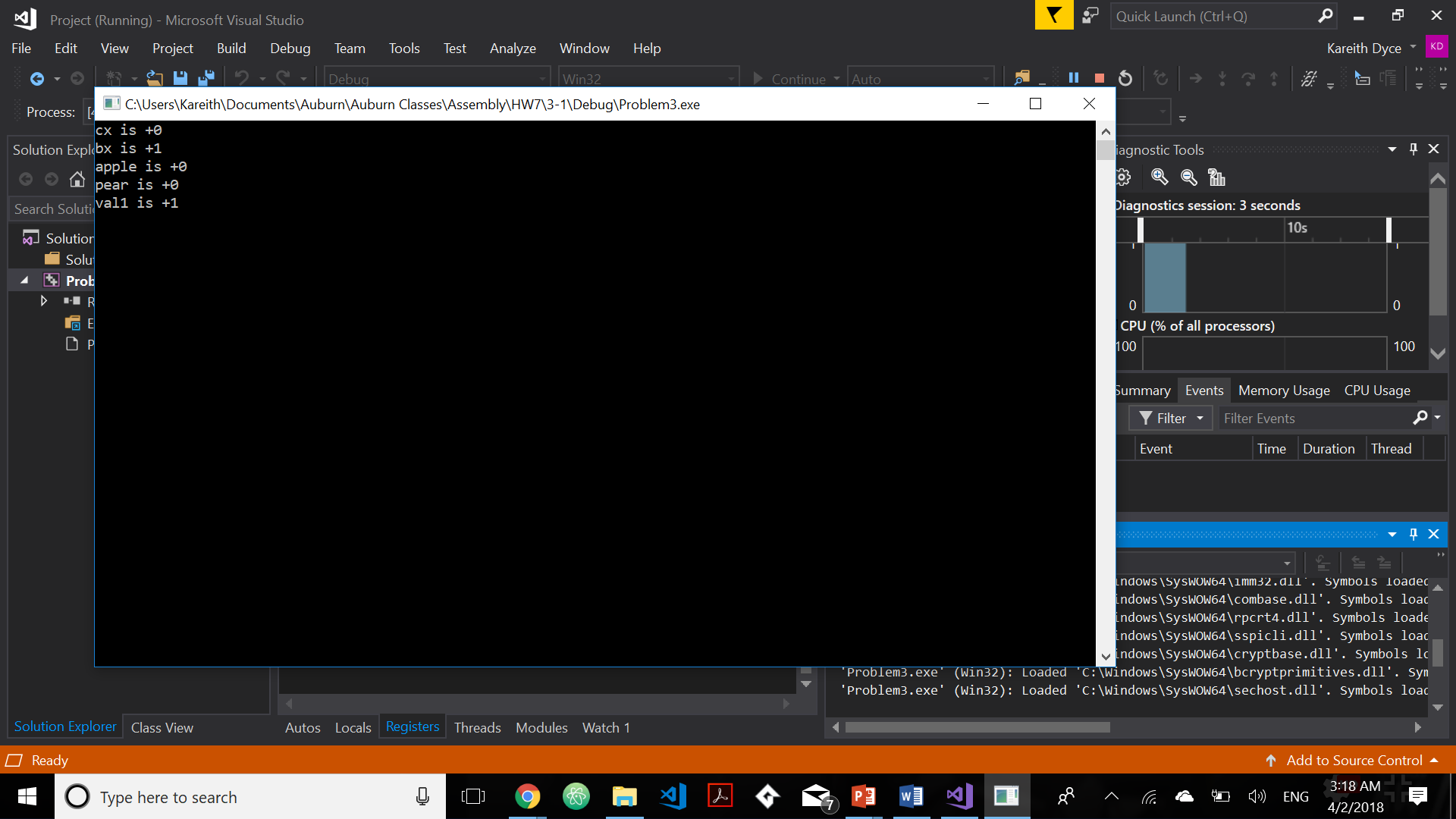
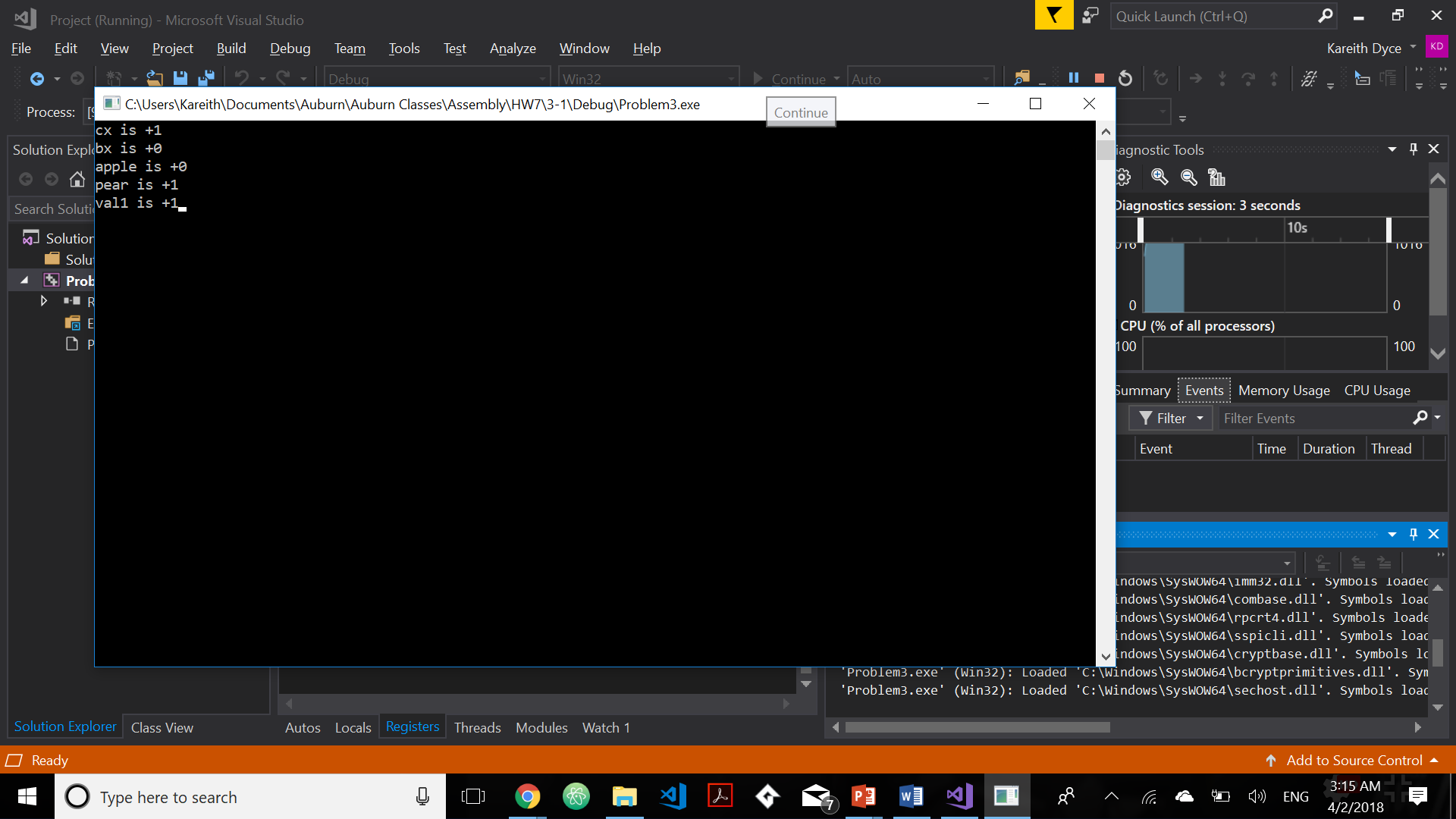
**Input test data**

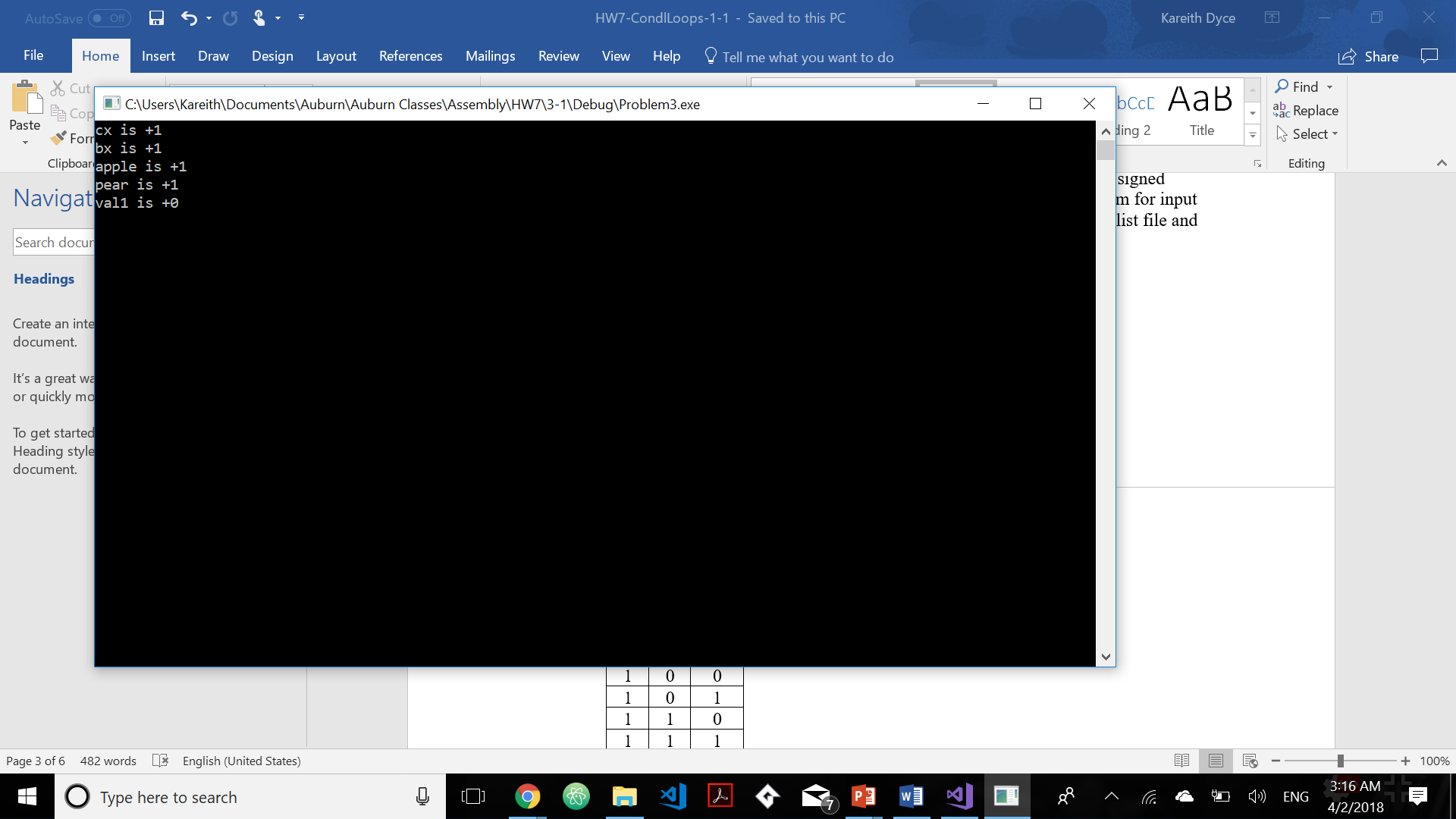
|  |  |  |
| --- | --- | --- |
| **CX** | **BX** | **Val1** |
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

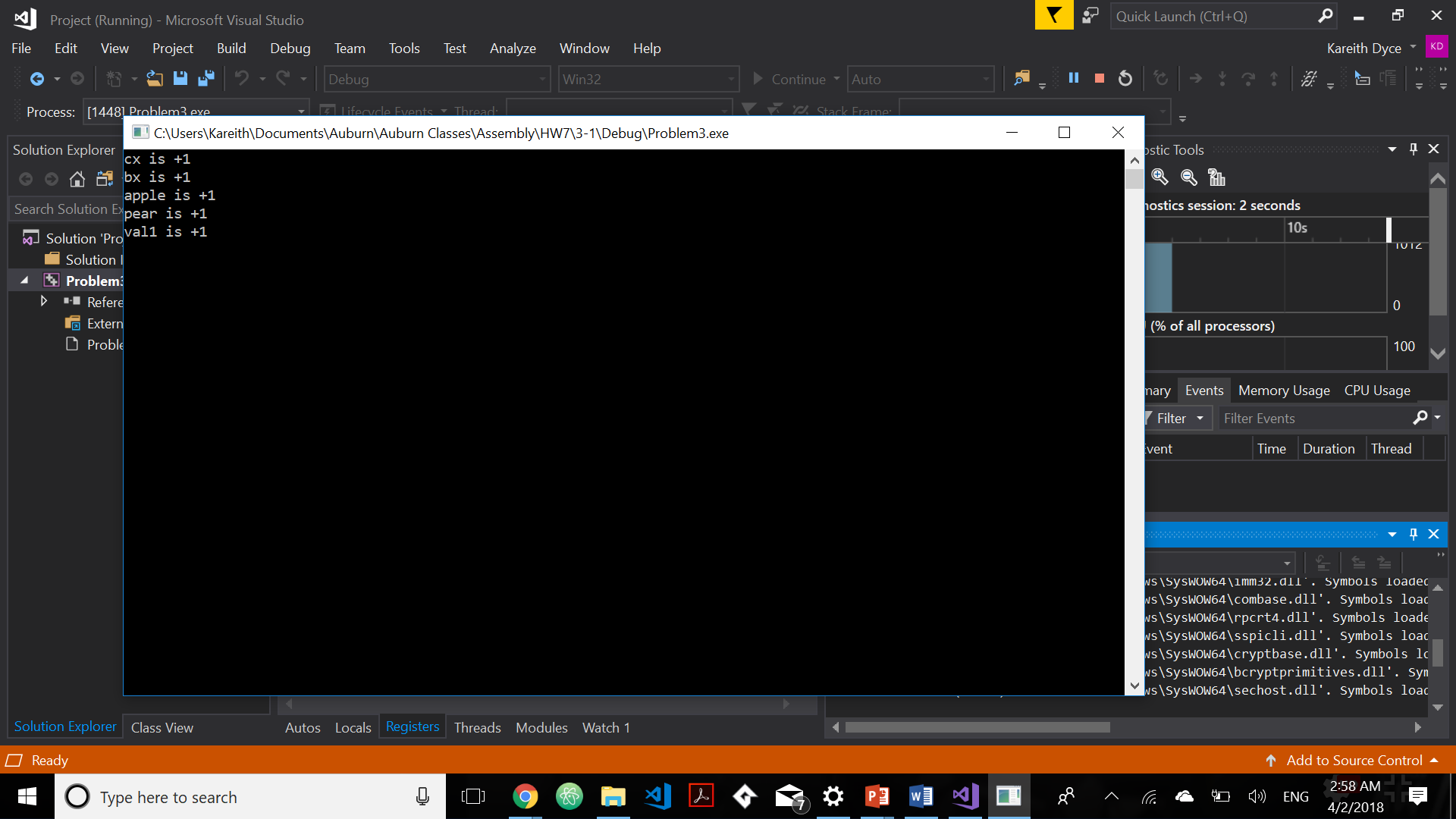












1. Draw the stack (pencil-paper or word🡪pdf) at different points of the main and subroutine to show your understanding of the call and return functions.

Main PROC

4040040 call FloatAdd

4040046 mov eax, ebx

…

…

FloatAdd PROC

4041020 Push ecx

4041024 Push ebx

4041028 mov eax, edx

…

…

404A030 Pop ebx

404A032 Pop ecx

404A034 ret

FloatAdd ENDP

