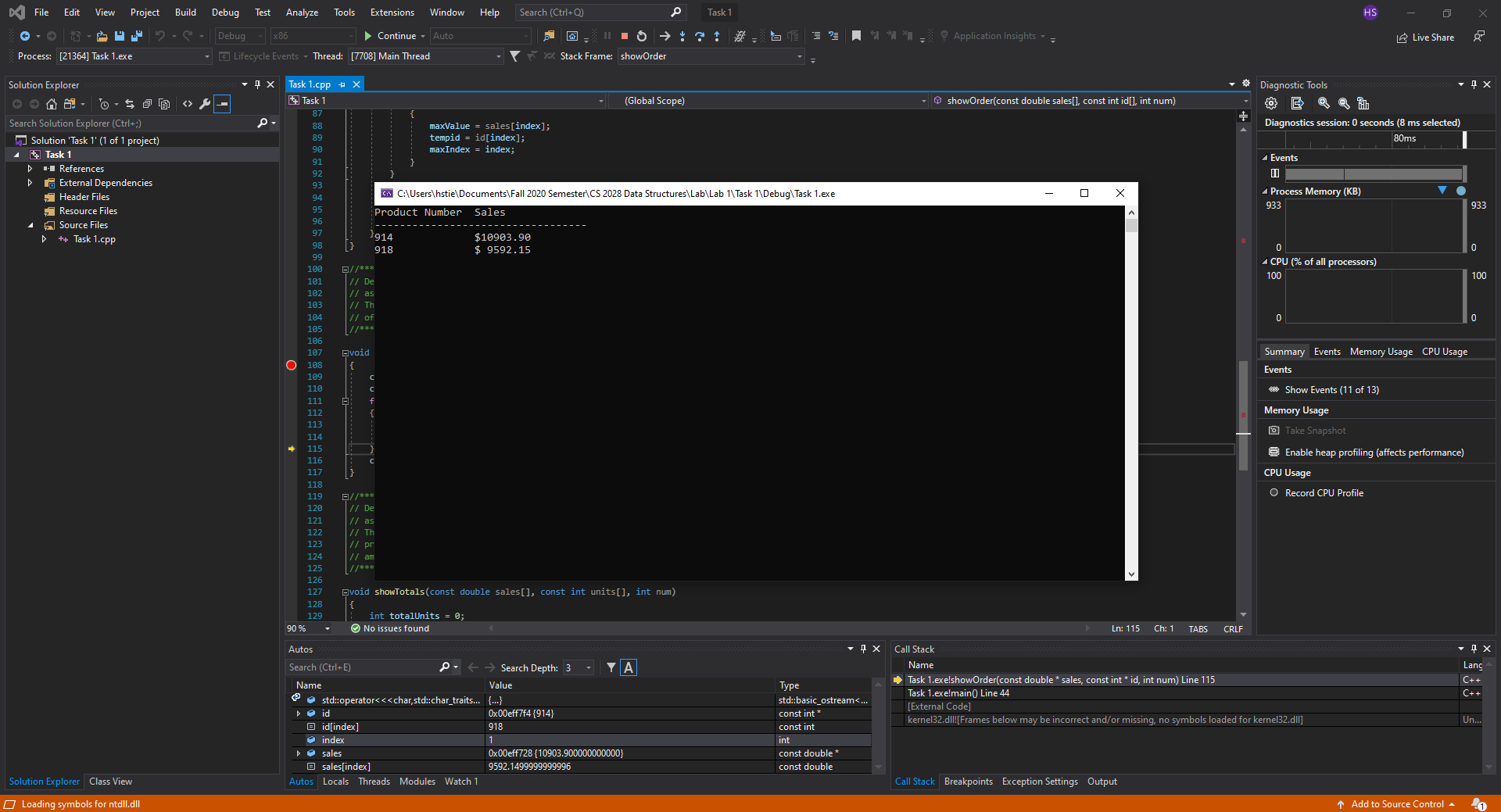
Henry Stiefel

Data Structures (CS 2028C)

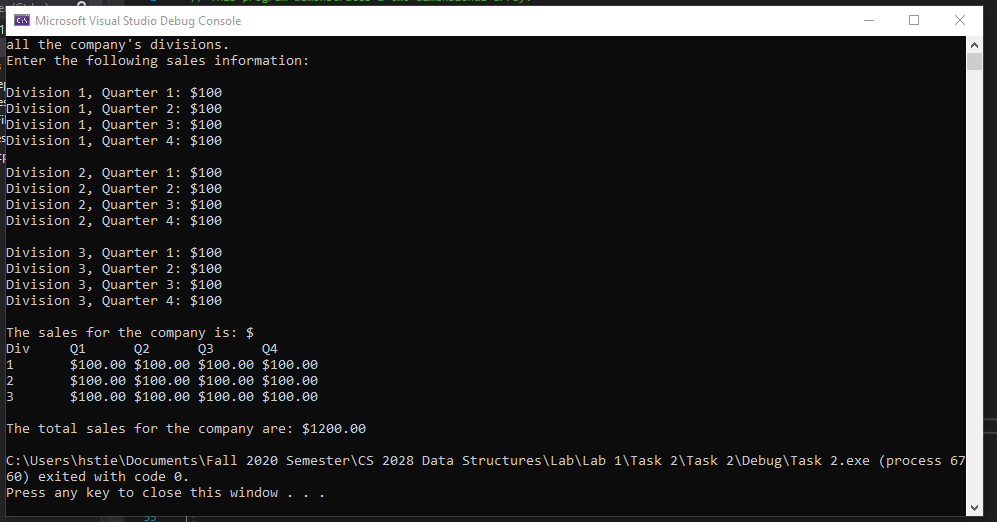
Lab 1

Task 1



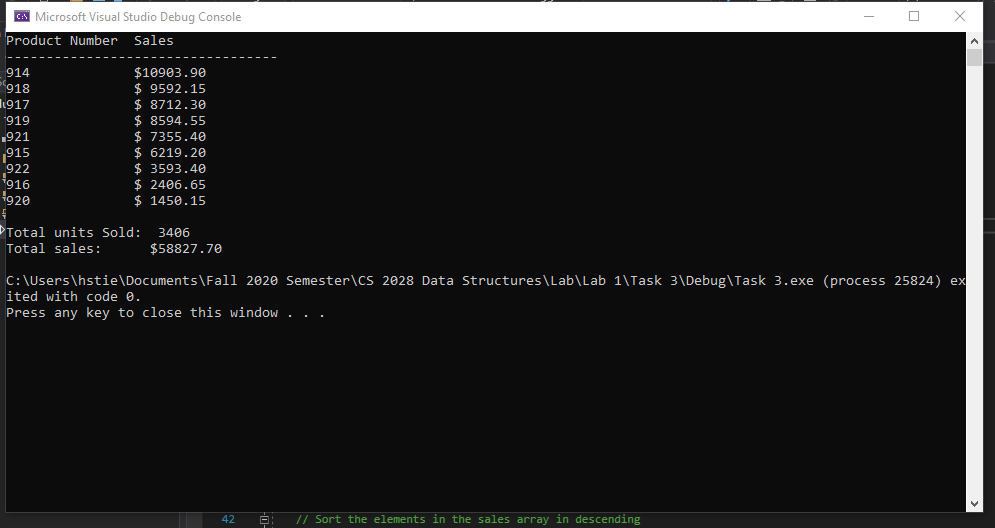
The objective of this task was to become familiar with the process of using breakpoints to test and debug code. This is important because being able to identify issues with projects and knowing exactly where to look to fix them is crucial to developing an efficient, working solution, and breakpoints allow you to iterate line by line to view code performance as it is happening.

Task 2



Our approach to debugging Task 2 began by running the code and examining the output for anything that appeared to be incorrect or unusual. After noticing several weird output strings and formatting issues, we took a closer look at the code to pinpoint what was missing or needed to be fixed. We believe these mistakes may have been made because the programmer was in a rush and simply forgot to double check the code to ensure the output was correct. They were very simple mistakes, like switching up rows and columns in a 2D array or forgetting to add parts to an overall total. We believe having thorough, developed pseudocode to use as a blueprint for writing the C++ code would help to avoid these mistakes.

Task 3



In Task 3, we had to create an array of structures and modify the code so that each function would run utilizing this array. We began by creating our structure blueprint, called Product, with four member variables: ID, units, prices, and sales. Next, we declared and initialized a total of nine object of type Product, using the values already provided from the original code. At this point, the methods that had previously been written used the original arrays that were replaced by the structure – so we had to modify the parameters of the methods and slightly change the body code to ensure it would work with our structure. Rather than needing to send multiple arrays as arguments, the functions could take one argument which was the Product structure by itself – since the values that were originally in the arrays were now member variables of the structure. After changing this, the function prototypes needed to be adjusted to match the correct parameters, and then the actual calls to the functions needed to be changed. This fixed any bugs that may have been introduced using the structure and produced the same output as the original code.