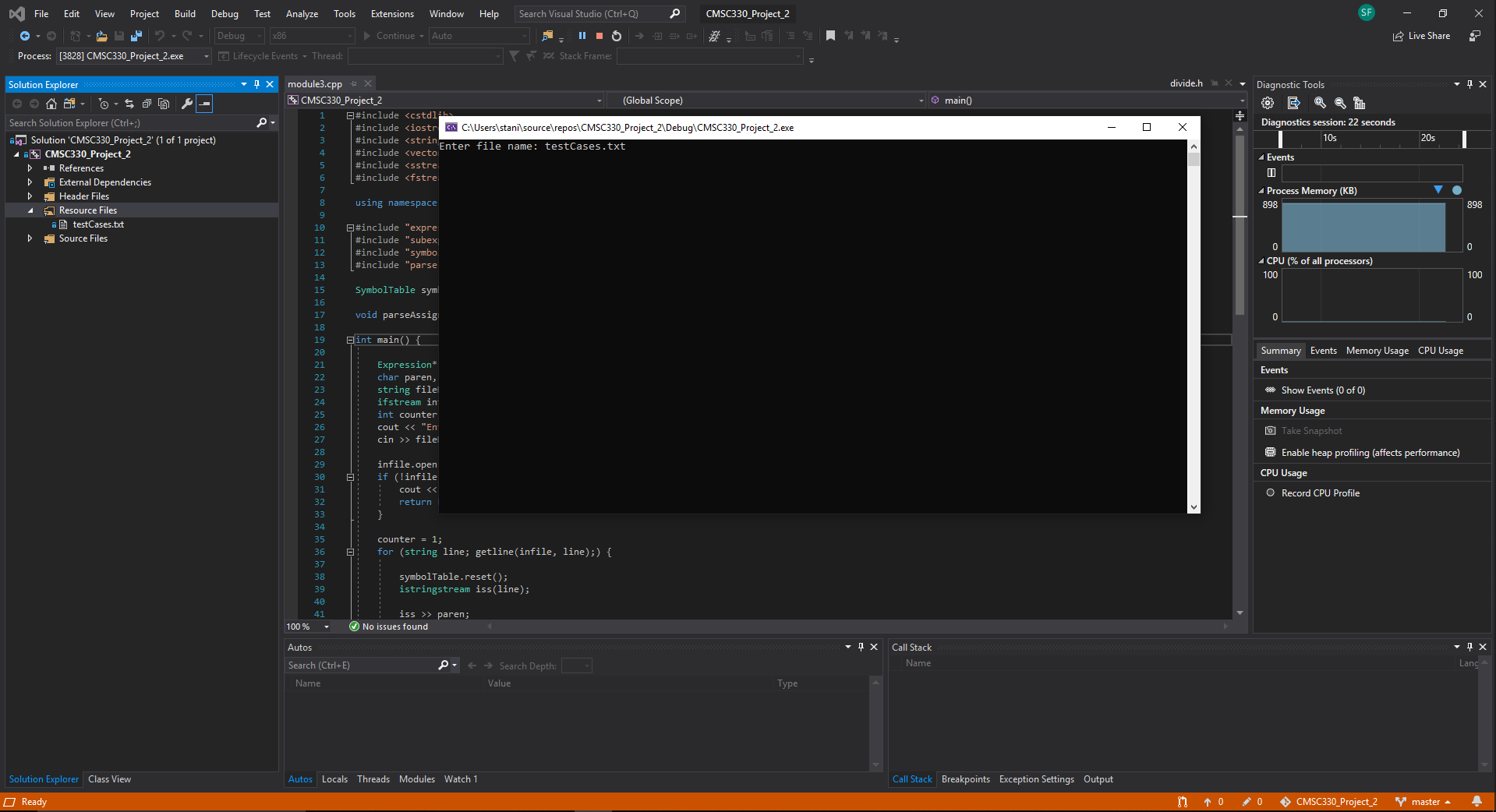
Stanislav Fatkhutdinov

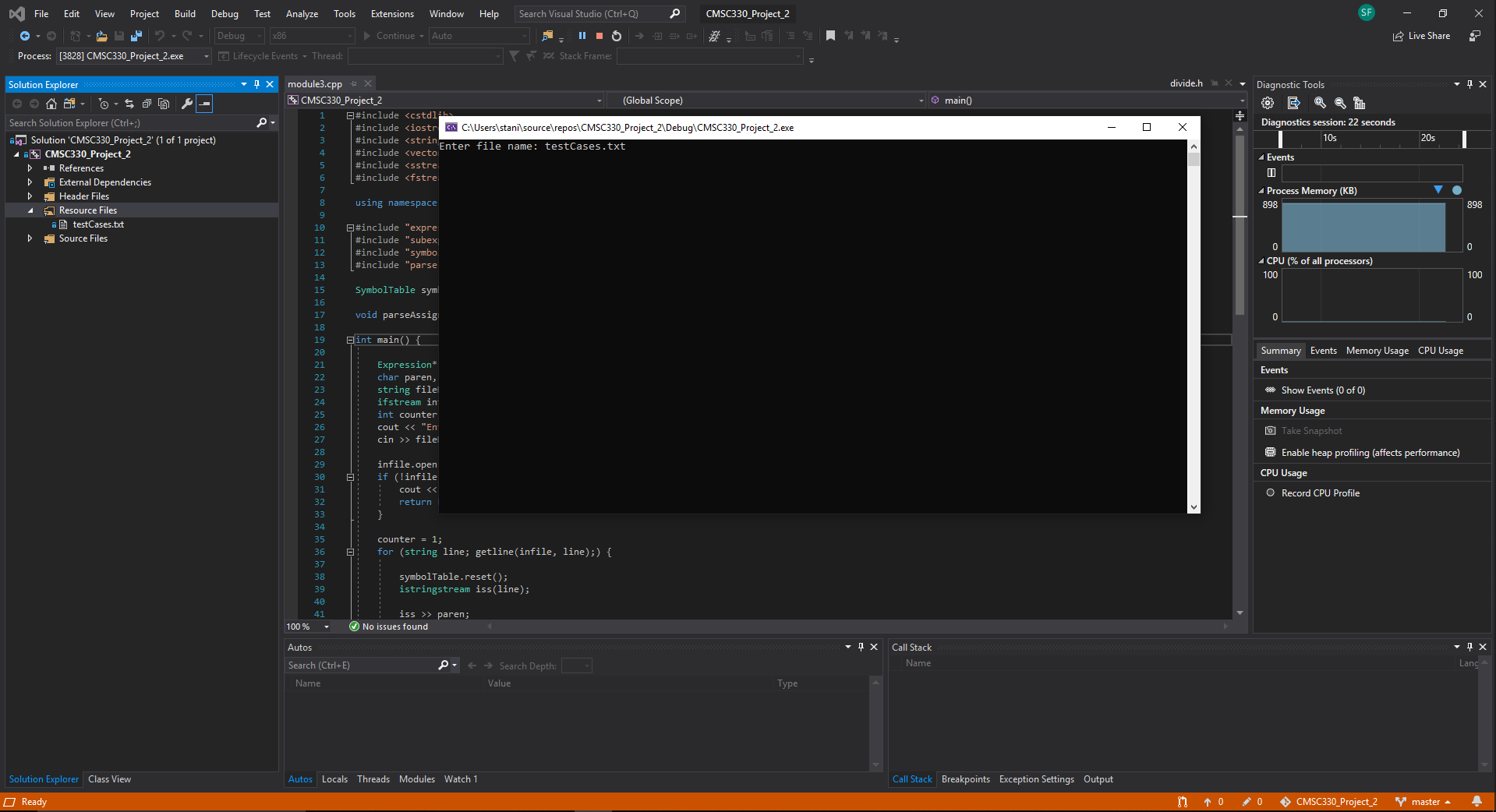
CMSC330 Project 2

The statements of the expression language consist of an arithmetic expression followed by a list of assignments. Assignments are separated from the expression and each other by commas. A semicolon terminates the expression. The arithmetic expressions are fully parenthesized infix expressions containing integer literals and variables. The valid arithmetic operators are +, –, \*, /. Tokens can be separated by any number of spaces. Variable names begin with an alphabetic character, followed by any number of alphanumeric characters. Variable names are case sensitive.

The program reads in the arithmetic expression and encodes the expression as a binary tree. After the expression has been read in, the variable assignments are read in and the variables and their values of the variables are placed into the symbol table. Finally, the expression is evaluated recursively.

Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Actual Output | Pass? |
| The program compiles and the window opens |  |  | yes |
| Inputting testCases.txt file  First expression  (x + (y \* 3)), x = 2, y = 6; | The file is read the output should equal to 20 |  | y |
| Expression #2: (x | (y & z)), x = 0, y = 1, z = 1; |  |  | y |
| Expression #3: (x < y), x = -2, y = -1; |  |  | y |
| Expression #4: (x > y), x = 5, y = 6; |  |  | y |
| Expression #5: (x = y), x = 5, y = 5; |  |  | y |
| Expression #6: (x : y ? z), x = 1, y = -1, z = 0; |  |  | y |
| Expression #7: (x!), x = 0; |  |  | y |
| Expression #8: (((x : y ? ((b | c) | (a!))) / (x : y ? ((a > b) & (c < d)))) - y), a = 1, b = 2, c = 3, d = 4, x = 5, y = -5; |  |  | y |

1. 

I’ve never really worked with C++ and it was a steep curve to switch from Java and Python for some reason for me. I watched a lot of youtube tutorials about creating C++ projects to complete this project. I might continue learning C++ after this class to better my understanding of the language.