SIM UOW CSCI251

ASSIGNMENT 1

Last updated: 8 January 2024

Timothy Mah

10258663

Lecture Group L01

Contents

[Introduction 2](#_Toc155625895)

[Limitations 2](#_Toc155625896)

# Introduction

Hello this is my documentation for SIM UOW CSCI251 ASSIGNMENT 1.

# Run

To install

sudo apt-get install libncurses5-dev

To compile

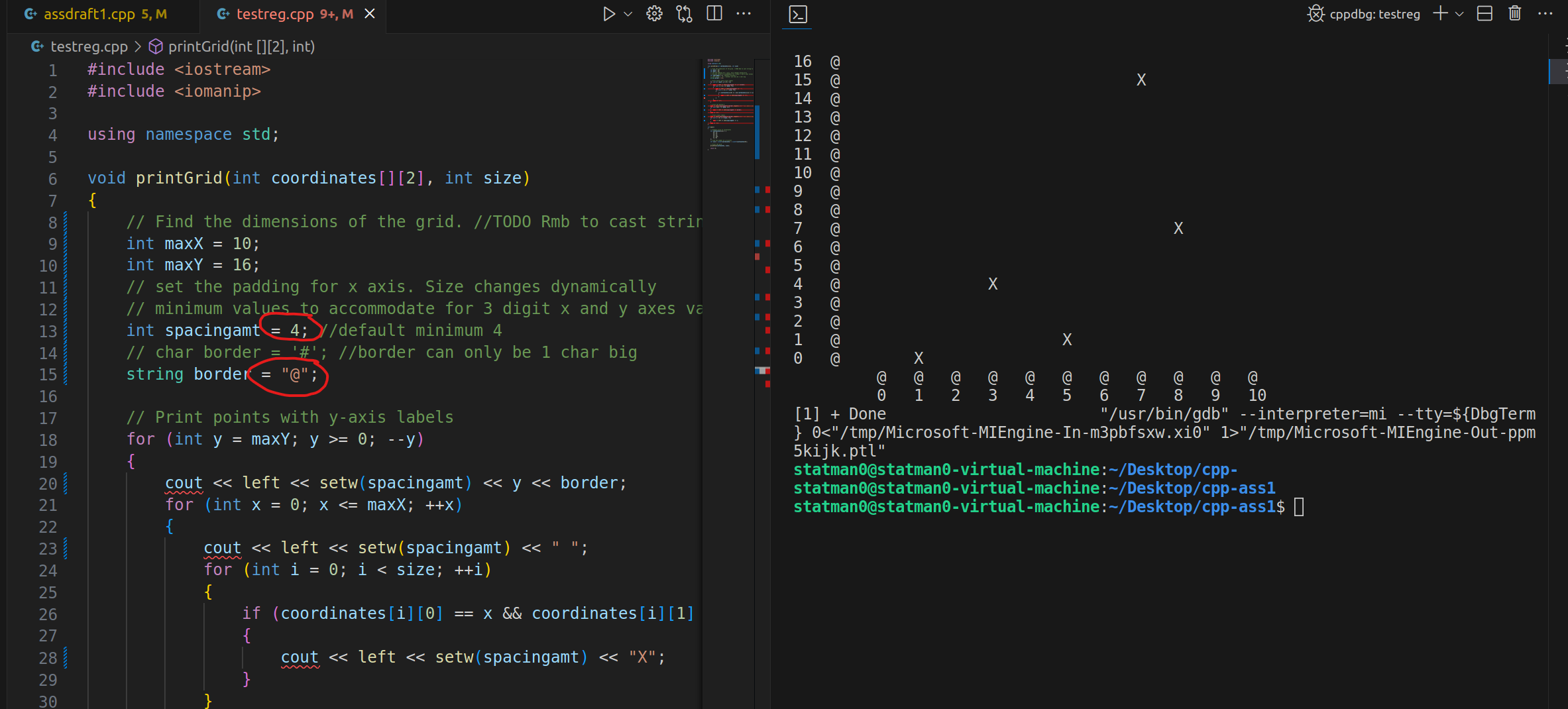
g++ testreg.cpp -lncurses -o testreg.app

# Extra features

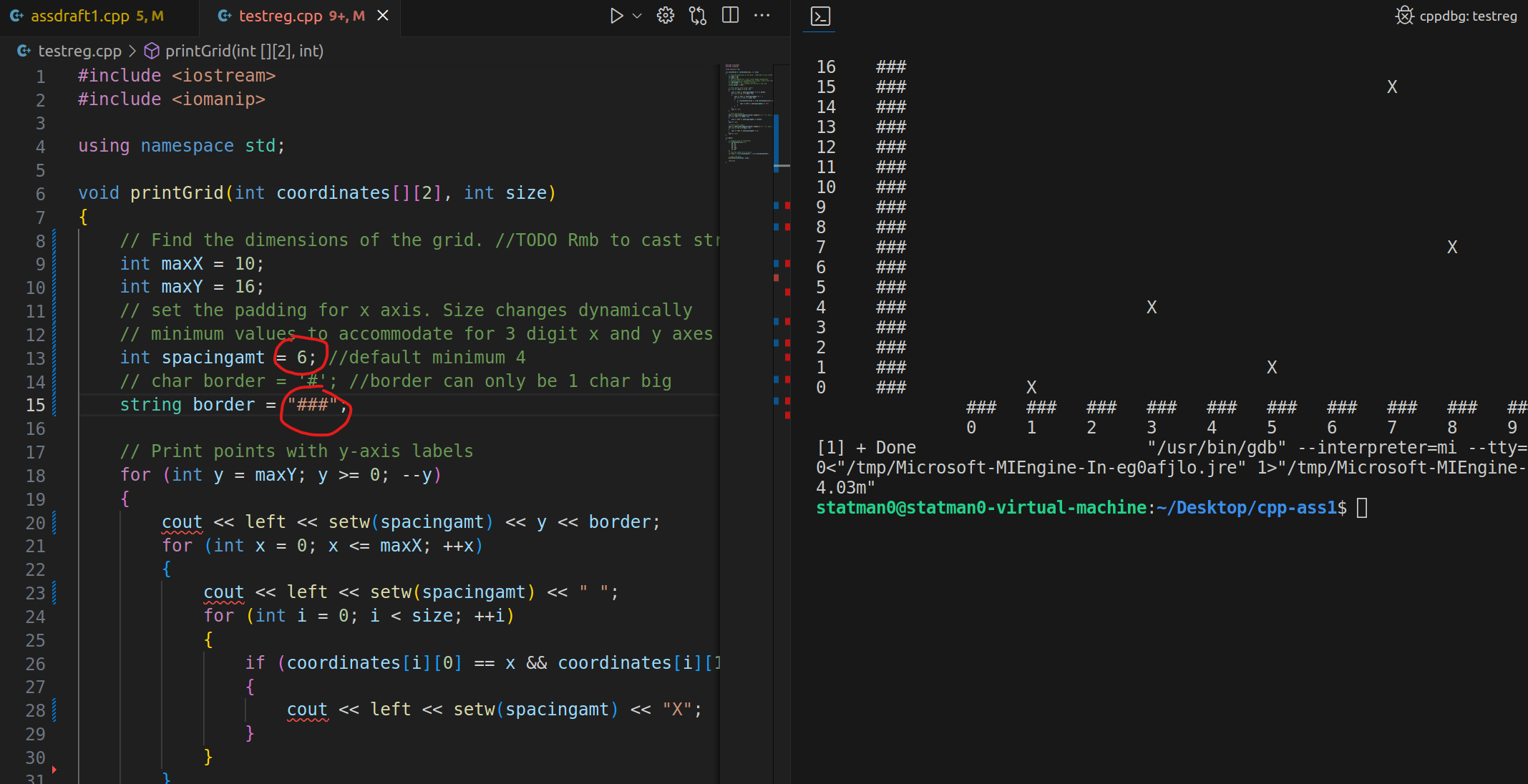
Hopefully these features give me extra marks ☺

## Dynamic border size and grid spacing

I have written my grid algorithm such that the # border can be set to any string, and any character. But if it is more than 4 characters long, the grid starts to look a bit crazy. The grid can also be dynamically adjusted to be horizontally wider or narrower. Wider grid is easier to read while a narrower grid can fit more things within the same terminal.



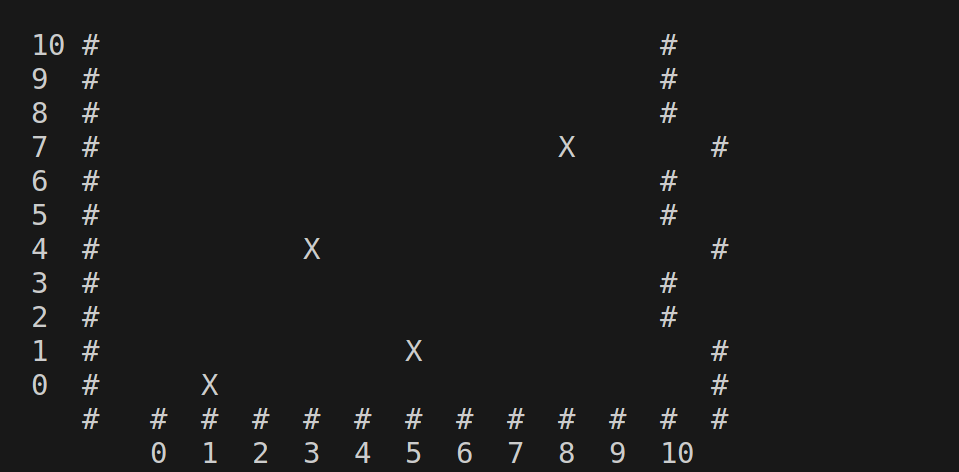
Above is a grid with spacing of 4 and a @ character border.



Above is a grid with spacing of 6 and a ### string border.

As illustrated, the change in border and grid size does not affect the X markings in the grid. The x and y coordinates remain consistently accurate.

# Limitations

1. Regex string may be system agnostic. However, in today’s context where many programs run over the web, server side processing would mitigate this issue.
2. The ncurser library is only available on Linux. But I stopped using it
3. The x and y axes of the grid can only be 2 digits big. Any bigger and the printing would have issue. But I think it’s fine because if the x and y axes are 3 digits big, the issue won’t just be the printing. The issue is that the user cannot even see the whole grid properly in terminal because it’s too big.
4. The application assumes that the x and y axes are integers. If it’s a decimal number, the decimal values will be truncated
5. My right side border is senget due to some looping logic error. I have given up trying to fix it in the interest of completing the other functions. This is more of an aesthetic aspect so I believe the other functional features are more important
   1. 

# Learnings

Some cool stuff I learnt

## How to update elements in vector with range type for loop

Personally I prefer to code in python style. Hence, the python style for loop is more intuitive to me.

        vector<string> strings;

        for (string s: strings)

        {

            s = "some string";

        }

In the above code snippet, the strings within the vector would be updated with the value, “some string”. However, the value “some strings” will only stay persistent within this for loop. Outside of the for loop, the values in strings is still empty.

To make the value persistent globally, reference the memory location of elements within the vector.

        vector<string> strings;

        for (string &s: strings)

        {

            s = "some string";

        }

Just one simple & is needed.

Of course I could also do the traditional for loop style with counter++ and all but I like this python style for loop. So this is like a mix of C++ style with python. Lol

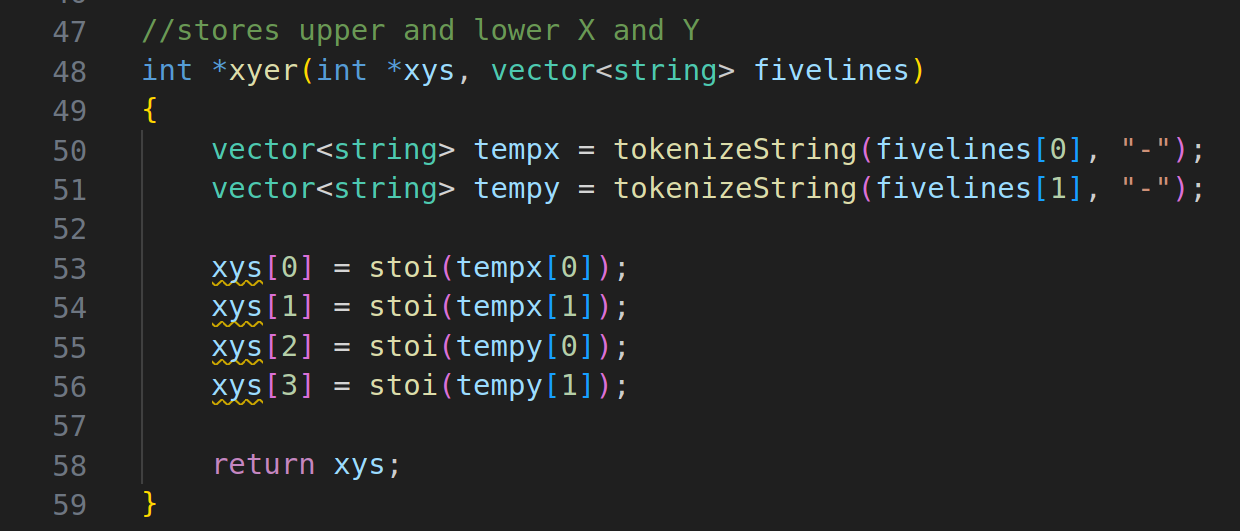
## Some functions want C style string, not regular string

I used this function called mvwprintw() from the ncurser library. It’s fourth parameter takes in a string. I had an int type variable that I wanted to use. I looked up ways to convert int to string, however I still received an error message. Upon closer inspection, the function wanted C style string const char\*. So I searched up how to convert string to C style string. In the end I mashed two functions together to become to\_string(gridy).c\_str()

But in the end I realised I cannot use external libraries so I scrapped it lol.

## CWE-416: Use After Free

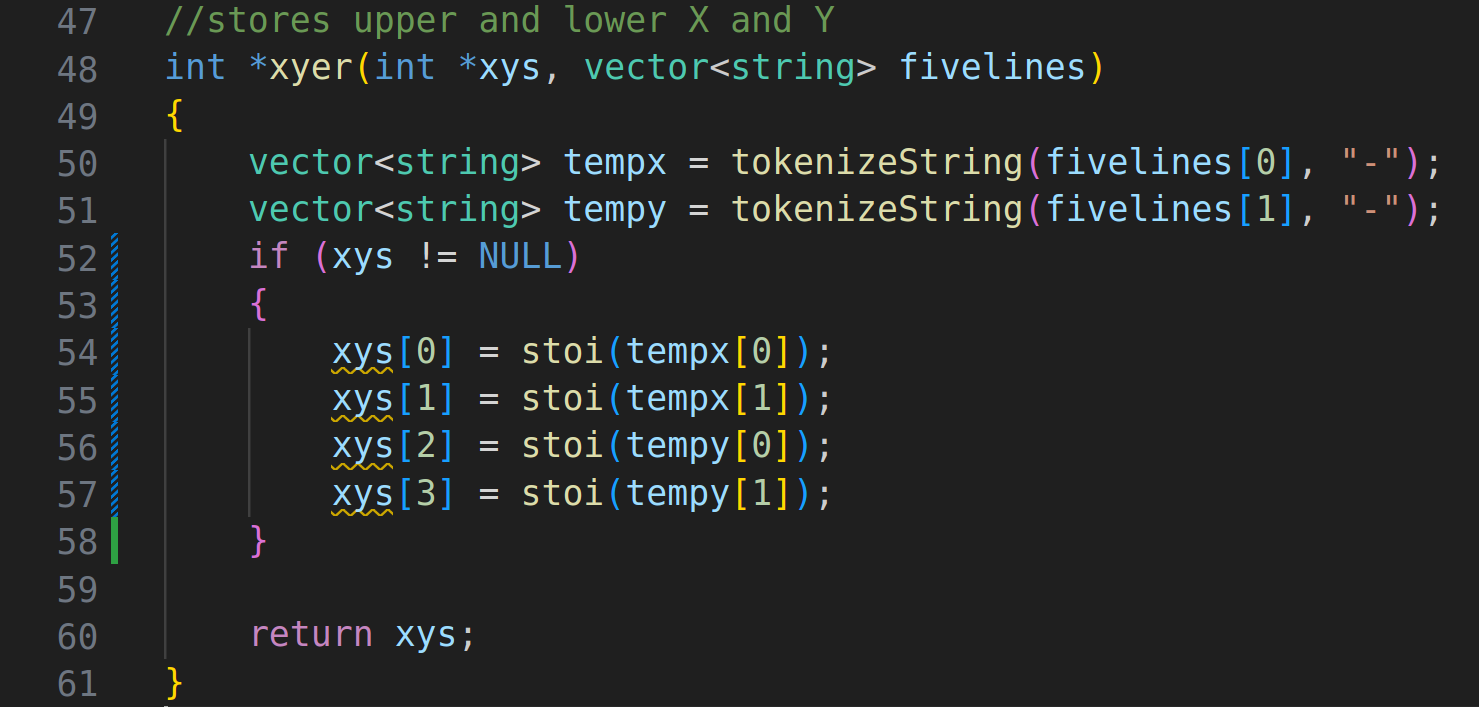
In my VS code IDE, I have a code scanning plug in from Snyk. It detected that my code had this vulnerability called “use after free”.



<https://learn.snyk.io/lesson/use-after-free/>

After reading the article I understood why Snyk was giving me a warning. I realised that if this function takes in an empty xys array, or if xys has a different number of elements than I expected, say only 3 elements, then xys[3] would be pointing to an unallocated memory location. Best case, xys[3] just points to some random memory location and when it is printed out, it just prints some gibberish. But worst case scenario is that an attacker could manipulate input to perform remote code execution.

Functionally my code as is would work because the size of the array is hard-coded but for good measure I should check if array is null before accessing its contents.



Nonetheless, the scanner still throws a warning. I think it’s fine because, as mentioned previously, the size of the array is hard-coded hence my app will not run into null pointer type issues in this function. Still, interesting to see Snyk giving me memory allocation related warnings. Something I would not see for python applications.