Lab 4 - Functions & Live Share

1. Warnings in VS Code

Create a `Lab4/` folder in your VS Code workspace and copy the `lab4.cpp` file (from course website) into the folder. Run the program to see what it does.

1. **The first few lines of `main()` have a bug; can you spot it?**
   - Run the program with different inputs for x; what do you notice?
     Hint: Try 42? What about 30? Should it do the same?
   - The code you ran is *valid* C++; however, there's a common bug in it, and the compiler can help us find it.

2. **Enable all C++ warnings:**
   - Open the file: `./vscode/tasks.json`
   - In the "`args`" section:
     add a comma on the last line,
     add the four highlighted lines shown on the right.

3. **Save `tasks.json`, switch back to `lab4.cpp` and recompile.**
   You may need to change the .cpp file's contents (such as adding a space) in order to be able to recompile.
   - If you encounter unexpected errors, such as unable to even compile when you press the run button, double check tasks.json (ensure there is a comma after every line except the last one).

```
{
  "tasks": [
    {
      "type": "shell",
      "label": "C/C++: g++ build active file",
      "command": "/usr/bin/g++",
      "args": [
        "-g",
        "${file}",
        "-o",
        "${fileDirname}/${fileBasenameNoExtension}",
        "-Wall",
        "-Werror",
        "-Wshadow",
        "-std=c++20"
      ],
      "options": {
        "cwd": "${workspaceFolder}"
      },
      "problemMatcher": [
        "$gcc"
      ],
      "group": {
        "kind": "build",
        "isDefault": true
      }
    }
  ],
  "version": "2.0.0"
}
```

Text 1: Contents of .vscode/tasks.json

4. Check out the compiler's warning. Does it help you find the bug?

   ```
   \CMPT130\Labs\Lab4\lab4.cpp: In function 'int main()':
   \CMPT130\Labs\Lab4\lab4.cpp:17:11: warning: suggest parentheses
   around assignment used as truth value [-Wparentheses]
        if (x = 42) {
            ~~^~~~
   ```

   - The compile suggests adding brackets: `if ((x = 42)) {…}`
     The warning tells us which line of code is the issues.
   - Note that this warning suggests a possible *fix* which would suppress the warning, but would not solve the bug. *(Since this is actually valid C++ code, the compiler is willing to accept it but suggests the added brackets if you really want this behaviour to tell future programmers its intentional).*

5. Edit the code and solve the problem! *(hint: = vs ==)*
6. **Every file you compile should compile without warnings.** Resolving compiler warnings greatly helps reduce bugs.
7. I suggest having your tasks.json file include the following args:
   `"-Wall"`: Turn on all warnings.
   `"-Werror"`: Treat all warnings as errors (i.e., you must remove them so it compiles).
   `"-Wshadow"`: Generate a warning if you reuse a variable name inside the same scope.
   `"-std=c++20"`: Use a modern standard for C++.

2. Making Functions

In the same `lab4.cpp` file (from course website) make the following changes.

Use functions to break-down the program into smaller pieces. Do not change what the program does (the output should be the same); just changing the program's *structure*.

- Which parts should be their own function?
  Hint: create one function for each calculation:

  Each of the calculations for n^2, sum 1..n, ...
  Think about the arguments these functions will need, and their return types.

  For example, create a function: `calcNSquared()`, *which accepts the integer* n *and returns the value of* n *squared (an integer).*
- Your program must have **NO** global variables; global constants are OK.
  - *Hint: If your function needs a value, pass it in as a parameter.*
  - *Hint: If your function computes something, return it.*
  - *Hint: If your function returns a value, make sure that the calling code (such as in* `main()` *) stores the result in a variable that you can use, such as:*
    `int my_answer = calcNSquared(5);`
- Note that for this exercises you will be making (very) short functions. Sometimes functions can be long (20-30 lines); sometimes they can be very short (1 line).
  Generally, shorter (1-10 lines) is best.

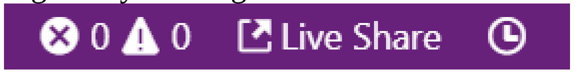
Understanding Questions

1. Explain why most function will accept values they need as parameters, such as:
   ```
   void printNStars(int n);
   ```
   instead of reading in the data from the keyboard directly in the function?
2. What is the difference between a function that uses:
   ```
   return 10;
   ```
   compared to:
   ```
   cout << 10;
   ```
   Which one will we usually use?

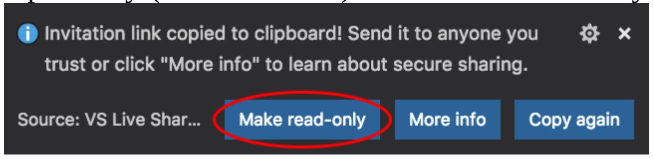
3. VS Code – Live Share (Optional)

You may **optionally** attempt this section if you are running Visual Studios Code, or Visual Studios. Live Share allows you to share your VS Code project with someone else for interactive coding or debugging.

1. You may want to watch a demo of VS Code's Live Share before continuing.
2. Create a new VS Code workspace named `LabLiveShare` (different than your current `cmpt130` folder). Note that this is not just a new folder inside your `cmpt130/` folder: it is a new whole workspace or folder at the same level of `cmpt130/`.
   - We will make a new one because Live Share shares the whole project and we don't want to share the entire project in case you have your assignment solution in it.
   - Make a folder on your computer named `LabLiveShare`
   - Open a new VS Code window and open this folder: File → Open Folder. Copy the `.vscode/` folder into new folder from the previous project.
   - Create a new `.cpp` file which prints something to the screen.
3. Setup (derived from this guide)
   - Install the Visual Studios Live Share extension: click extensions icon on left ( ) and search "Live Share"
   - Sign in by clicking the Live Share status bar (bottom left):



     I suggest signing in via a GitHub account, but you can create an account as needed! (Note that this is not managed by SFU and is outside of our control).
4. Find a lab partner
   - Have a friend (or find a new lab-friend in person/on Discord) get ready to join your session.
   - *Suggestion, post to Discord's "general" chat channel "LFG Live Share lab"*
5. Start a live share session:
   - Enter the Live Share tab on the left ( ).
   - Double click the "Start collaboration session" option in the top left.
   - Optionally (recommended) click "Make read-only" so other person cannot edit.



Invitation link copied to clipboard! Send it to anyone you trust or click "More info" to learn about secure sharing.

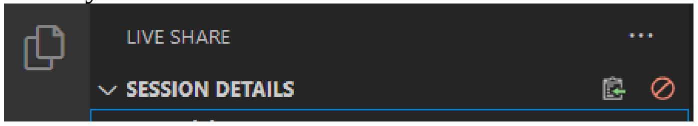
Source: VS Live Shar...

   - This will have copied an invite link to your clip board.
   - Paste the invite to your lab-friend. When they join you should be able to see the

details on the Live Share section on the left ( ).

Warning: Everything in your project will be shared to the other person, so be careful what you share (such as assignment code, etc).

6. Lab-friend Joins Session
   - Inside VS Code, go to the Live Share tab on left ( ).
   - Double click "Join collaboration session" and paste in the URL created above.
7. Experiment:
   - Run the program and show your lab-friend the output. Ensure they can also see your code.
8. When done:
   - Close your Live Share: On the Live Share tab on the left, click the cancel icon top left.



← **Click here**

   - Re-open your CMPT130 workspace: File → Open Recent → *(select CMPT130)*
9. Now, have your lab-friend send you a Live Share invite!

4. Submit

- Submitting your completed lab file to CourSys