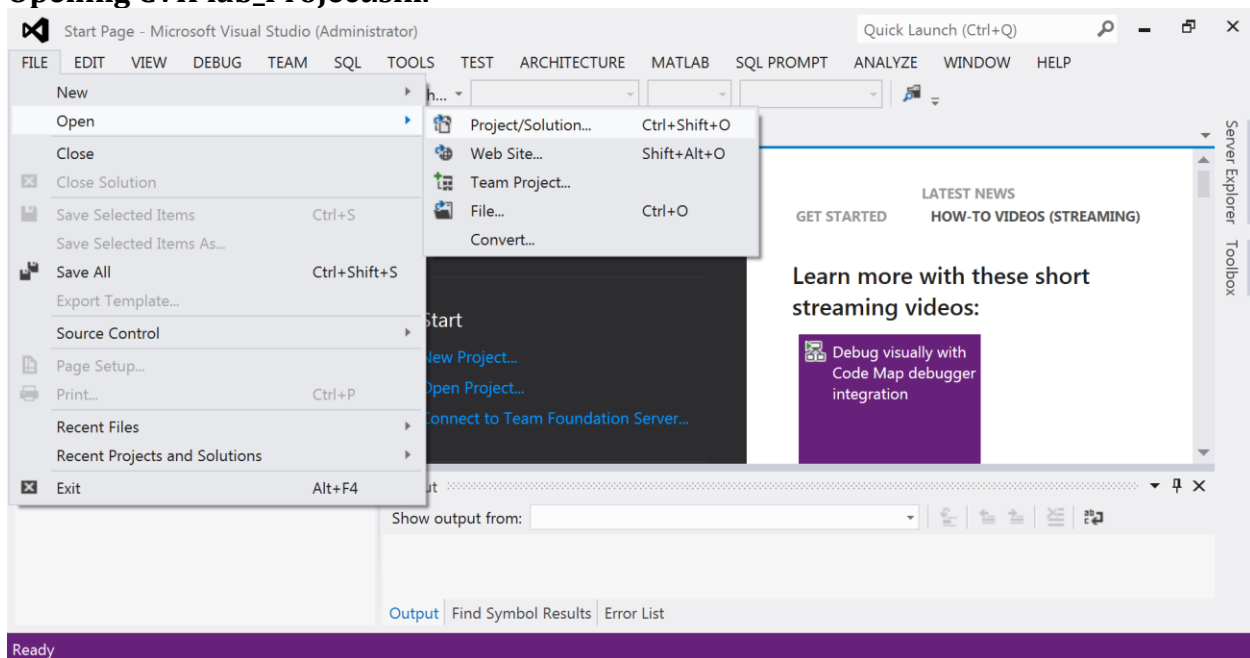


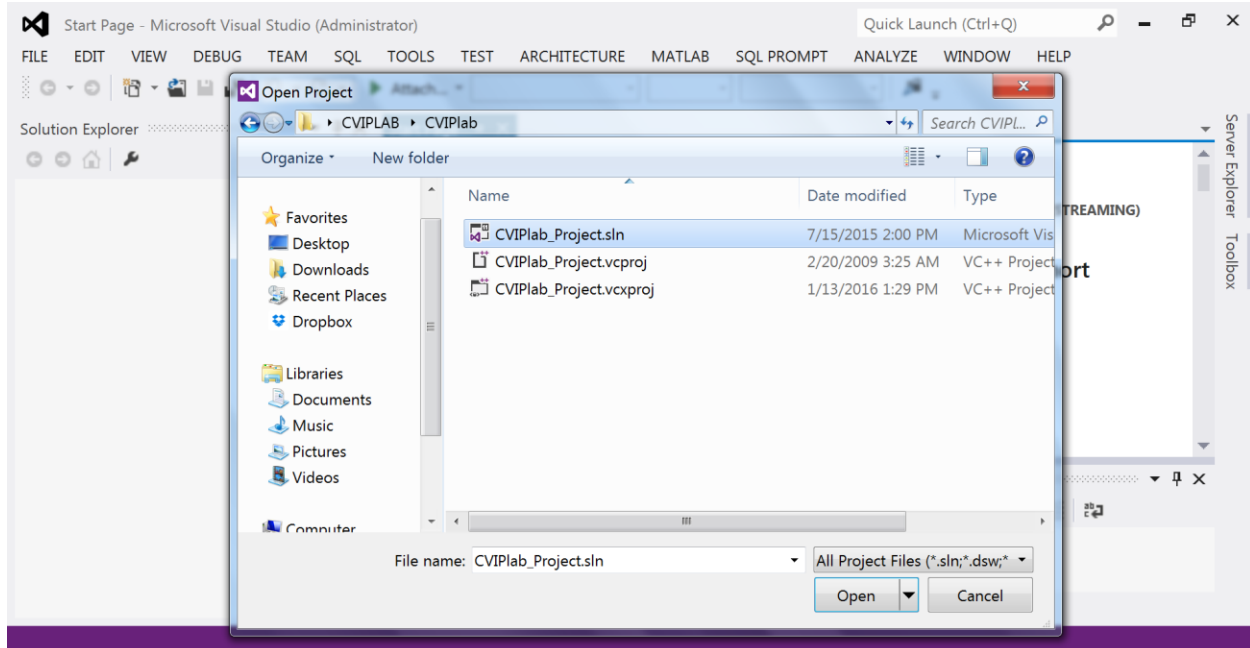
Compiling and Linking CVIPlab with Visual Studio

1. Install CVIPtools and CVIPlab
2. Choose the desired location for the installation – in this guide, we use C:\CVIPtools\CVIPlab as the working folder.
3. Run Microsoft Visual Studio.
4. Open the CVIPlab solution file, *CVIPlab_Project.sln*, in C:\CVIPtools\CVIPlab as shown:

Opening CVIPlab_Project.sln.



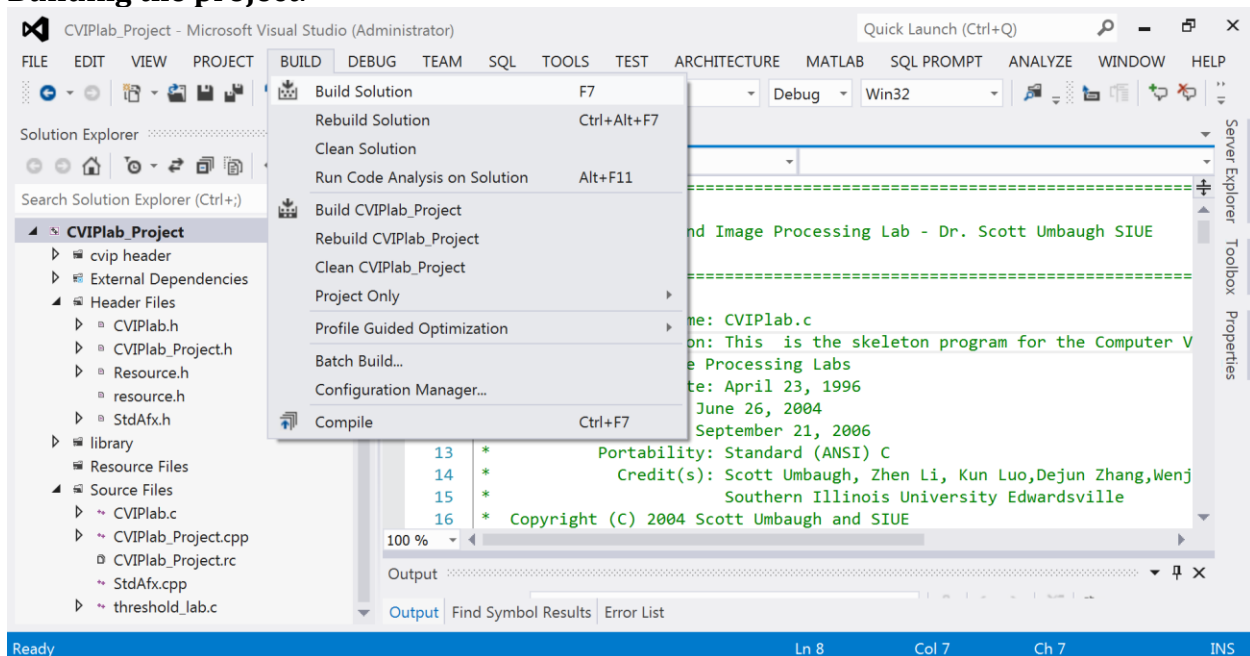
(a) Select *Open->Project/Solution*



(b) Select the file *CVIPlab_Project.sln*.

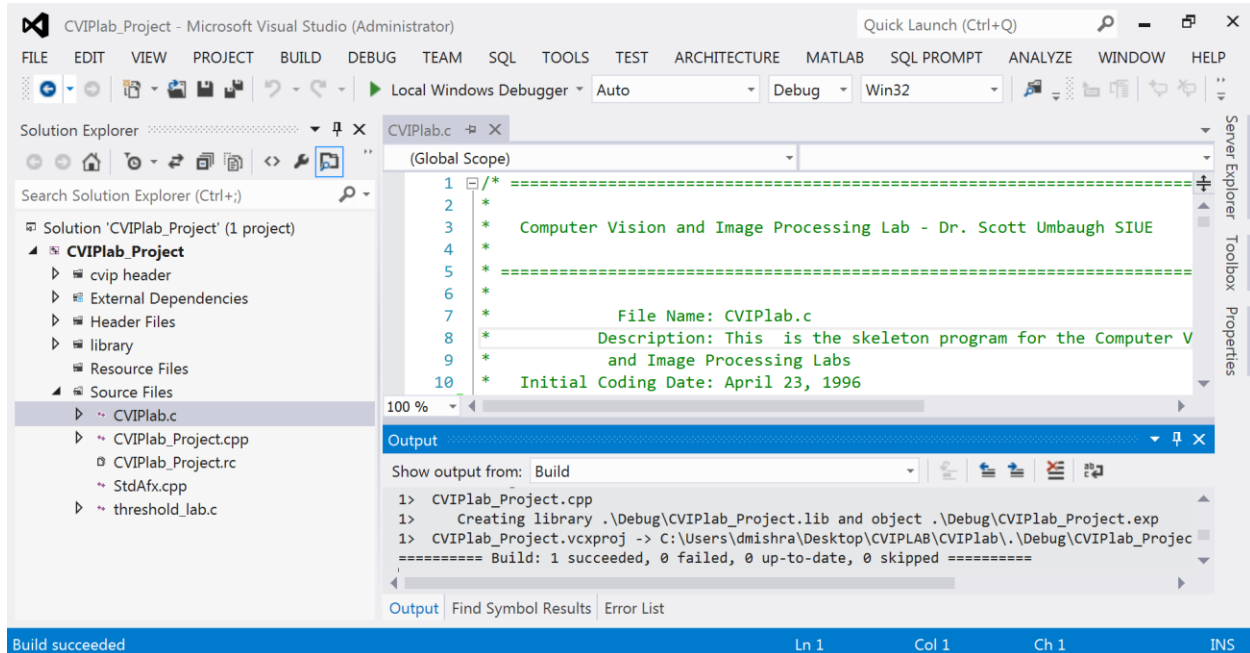
5. Build the project by selecting *Build* → *Build Solution* as shown:

Building the project.

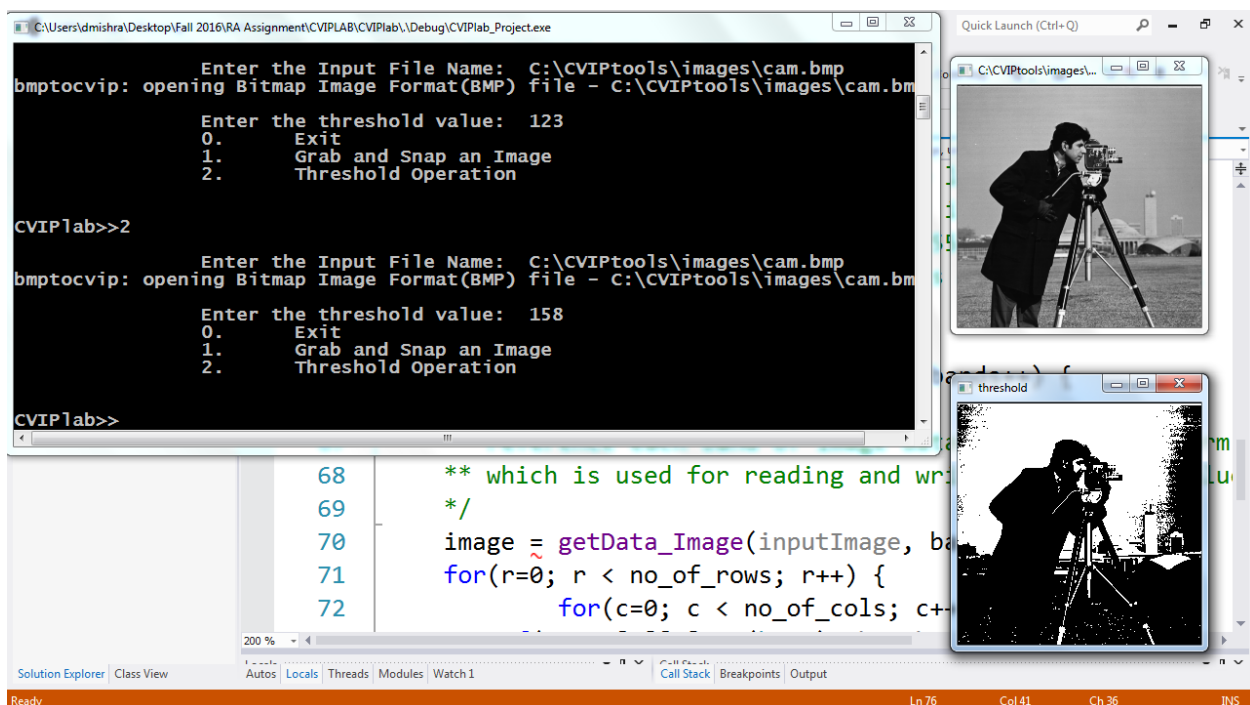


6. Activate the output window by selecting *Output* from the *View* menu, or press *Ctrl+W*, O (if it is not shown). *CVIPlab_Project* should be compiled with 0 errors as shown below in (a), and the executable file is located in *C:\CVIPtools\CVIPlab\Debug*. It should be noted

that it is not unusual to get warning messages during compilation. These warning messages should be investigated as they may indicate poor programming practices that can cause problems. In this case, the last few warning messages are due to variables that are not referenced, meaning they are not currently used in the program. Here, these variables are included for future use so we will not be concerned with these warnings.

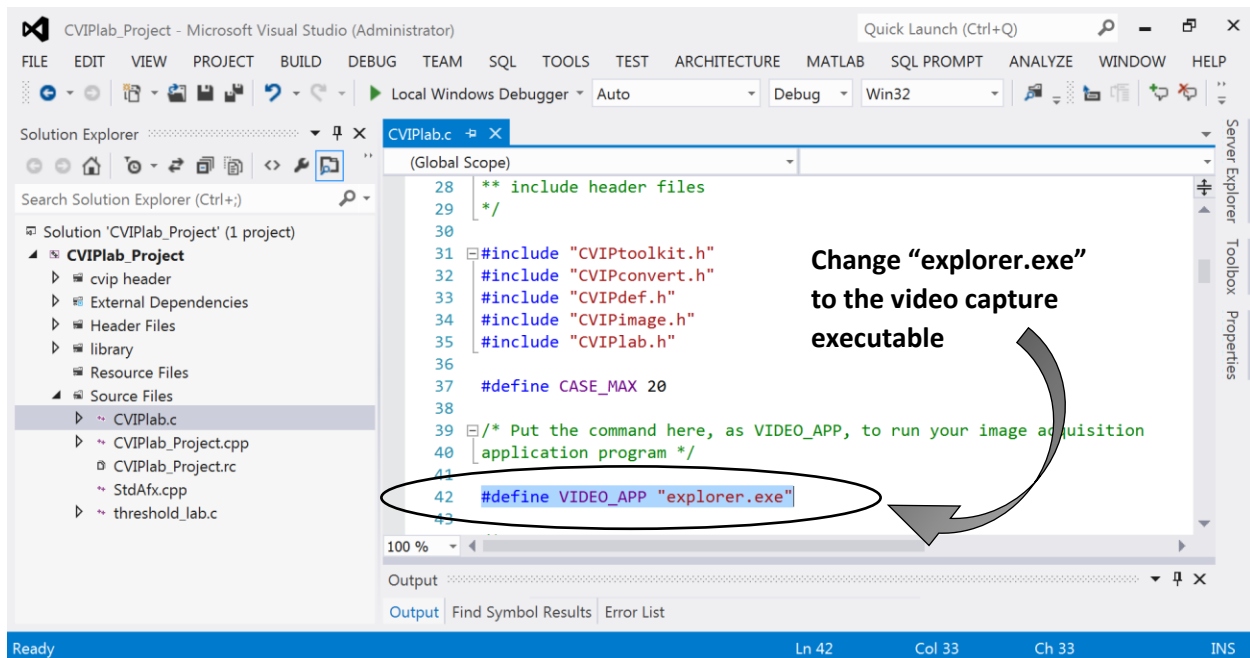


(a) Screen after compilation with no errors



(b) Screen after running CVIPlab and performing a threshold operation on cam.bmp. Note that CVIPlab requires a complete path name for the image files; or the images can be put in the same directory from which CVIPlab is running.

7. Press F5 to run the program. Select '2' and enter the file name for an image in the directory, or enter the full path name for an image elsewhere – here we used cam.pgm from the C:\CVIPtools\images directory. Enter a threshold value to perform the threshold operation, as shown in (b) above. If you see this, CVIPlab has complied and run successfully!
8. If the computer being used has video capture capability, add this to your CVIPlab as follows: Open file cviplab.c, go to the `#define VIDEO_APP` line directly after the include header section, modify the video/image capture string to the executable you plan to use, as show below:



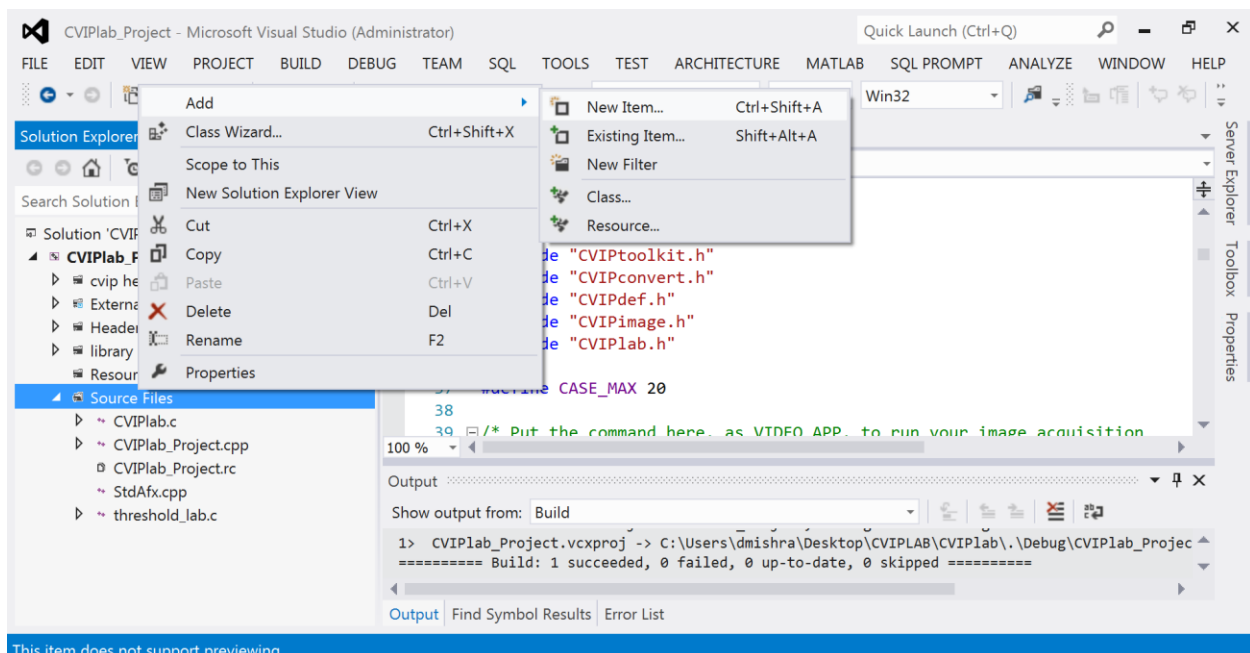
9. Run CVIPlab project by pressing F5. Select 1, and your video/image capture application will run.

The Mechanics of Adding a Function with Visual Studio

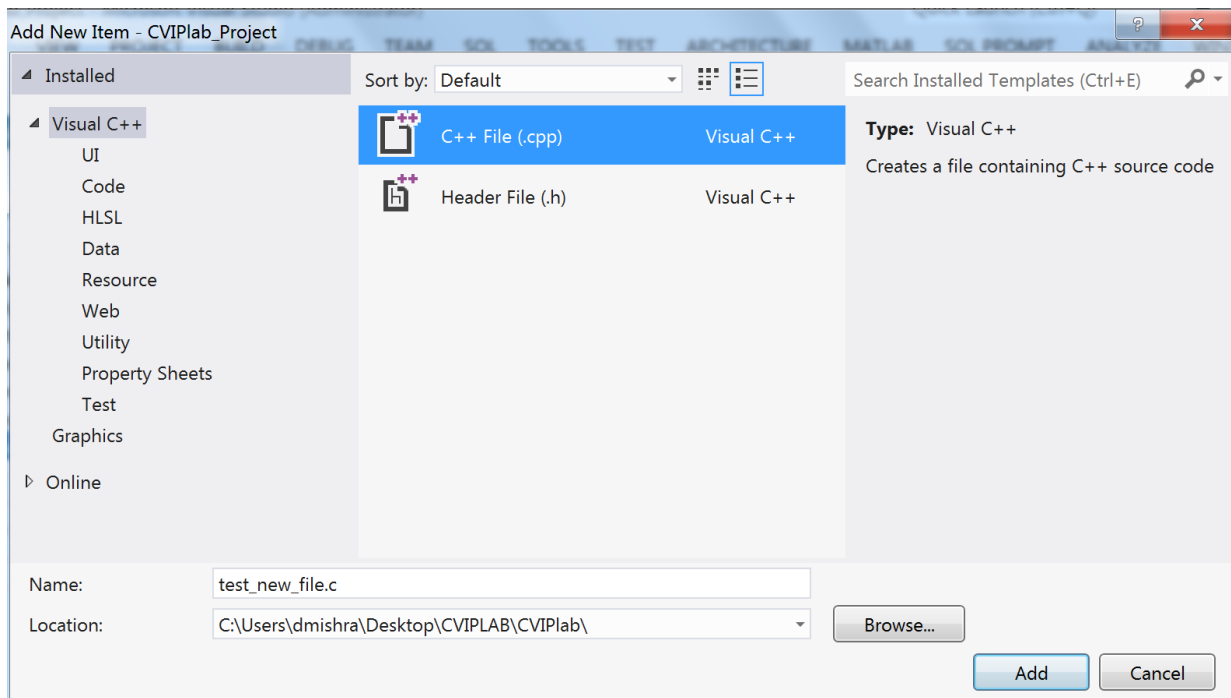
The following guide provides a step-by-step process for those unfamiliar with Visual C++. For those familiar with Visual C++, or those planning to use another compiler, skip to the next section, which provides details for adding a CVIP function to the CVIPlab menu.

To add a function using Visual C++:

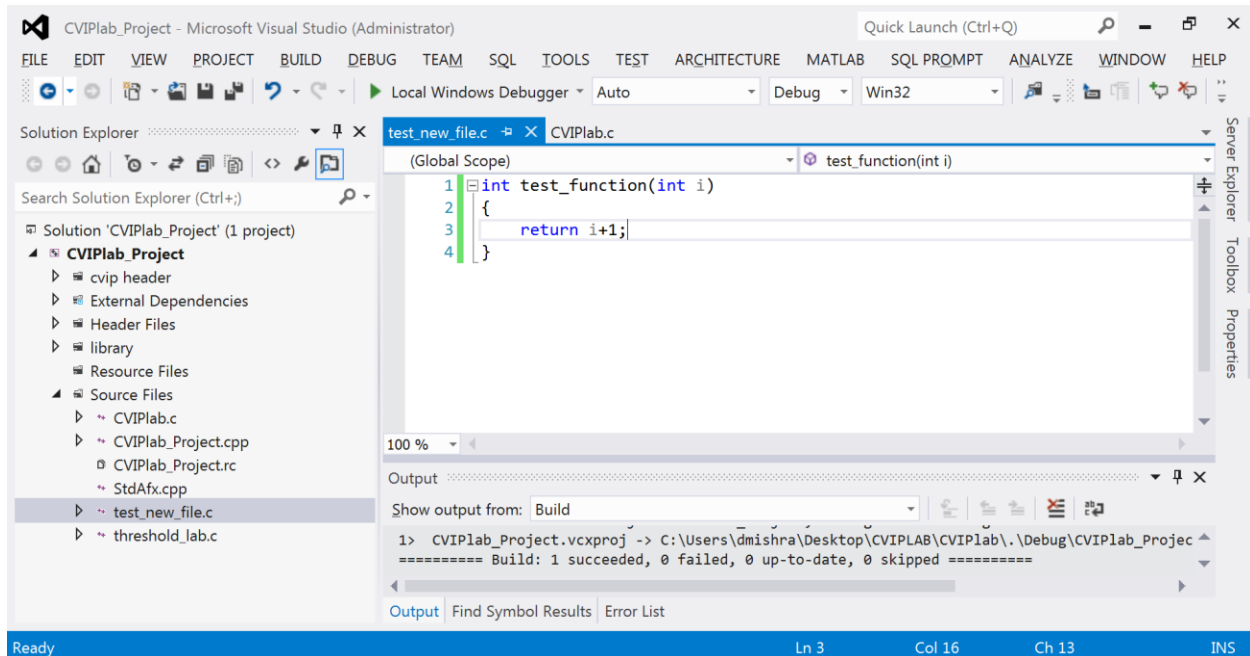
1. First add a new file by right clicking on the *Source Files* folder in the *Solution Explorer* window on the left. Then select *Add->New Item* as shown in (a) below.
2. From the upcoming window select C++ File (.cpp) and input *test_new_file.c* as the name, as shown in (b) below. Even though the C++ File option is selected, be sure to keep the extension “.c” for the new functions or files. Otherwise, compilation errors will occur.



a) From the Solution Explorer menu, right click on *Source Files* and select *Add->New Item*,



b) Select *C++ File (.cpp)*, as shown with an arrow, and name the new file “test_new_file.c” and press *Add* button



c) Select test_new_file.c from Solution Explorer menu and type in your function body.

3. Click *Add* button and enter the text below in *test_new_file.c*, as show in (c) above.

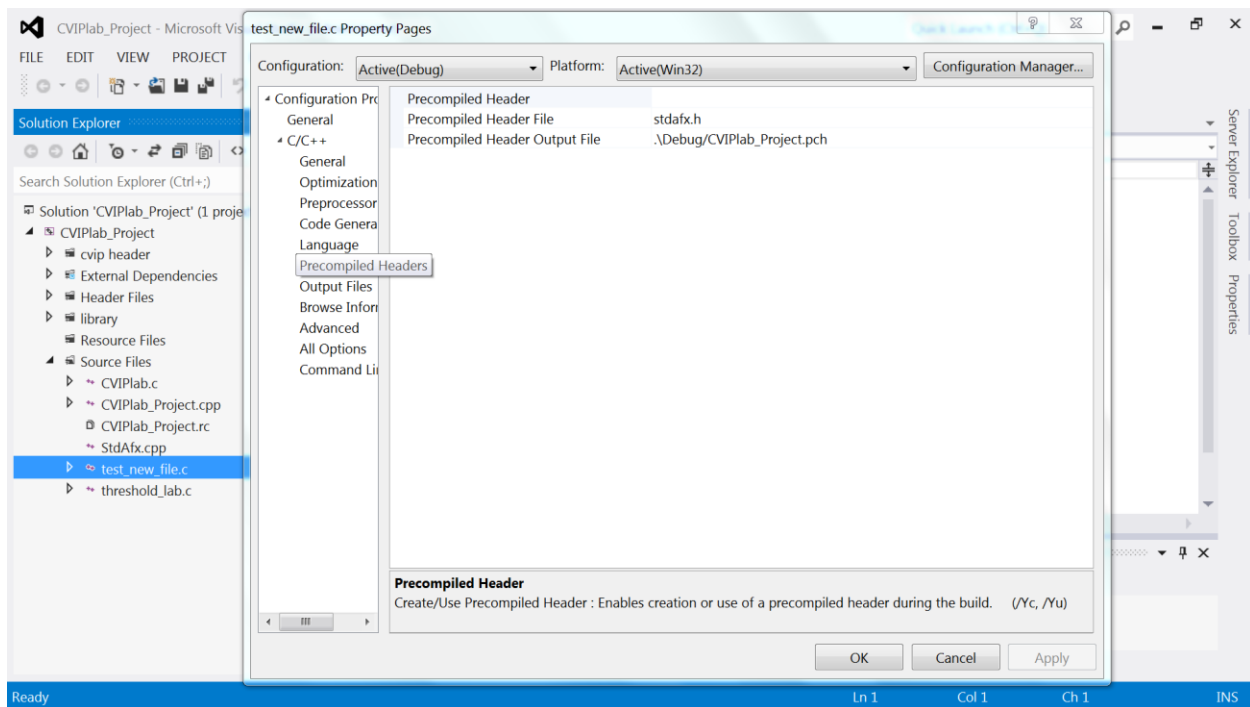
```
int test_function(int i)
```

```

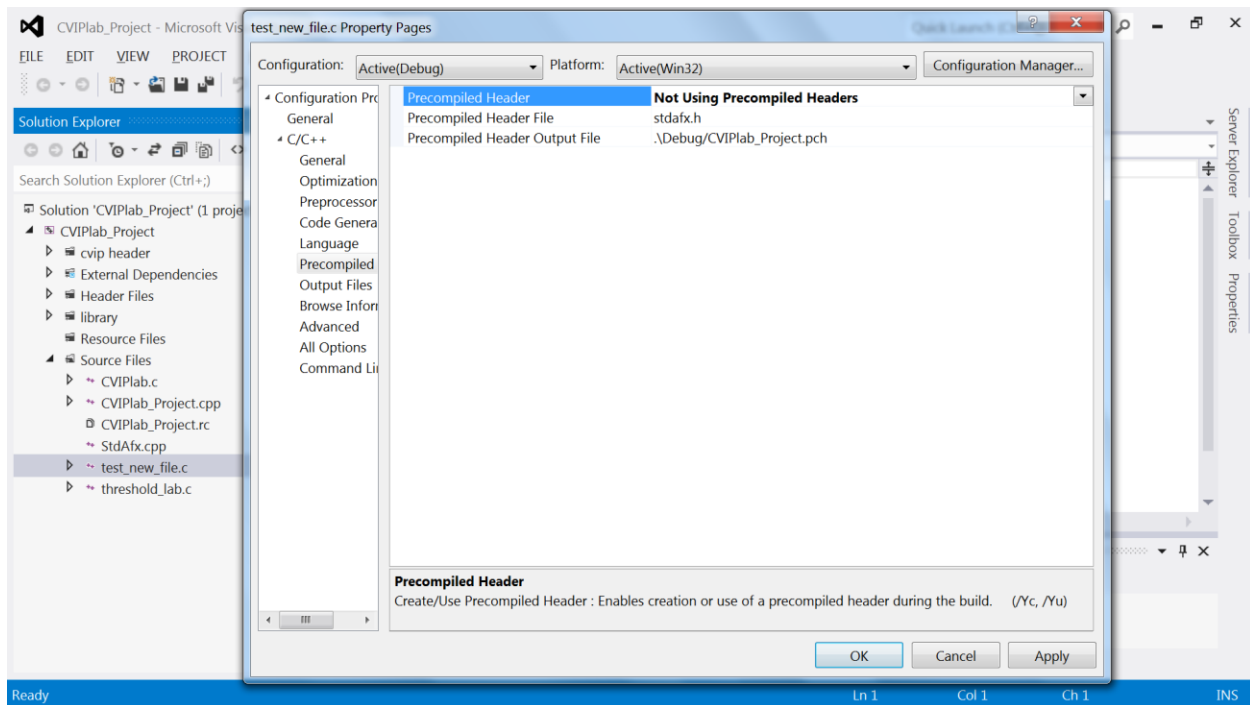
{
    return i+1;
}

```

4. Right click on *test_new_file.c* (located under CVIPlab_Project\Source Files) and select *Properties*. From the upcoming window expand the C/C++ selection on the left and select *Precompiled Headers* as shown below. Check the text box *Create/Use Precompiled Header* to make sure that *Not Using Precompiled Headers* is chosen. If it is chosen click OK, if not, click on the text box to change the selection.



a) Right click on *test_new_file.c* and select *Properties*. From the upcoming window, expand *Configuration Properties*, then expand *C/C++* and select *Precompiled Headers*



b) From the *Precompiled Headers* section, click the combo button next to the *Create/Use Precompiled Header*, and select *Not Using Precompiled Headers*

5. Select *Build->Build Solution* from the menu bar above, to compile the project again. There should be no errors in the output box.

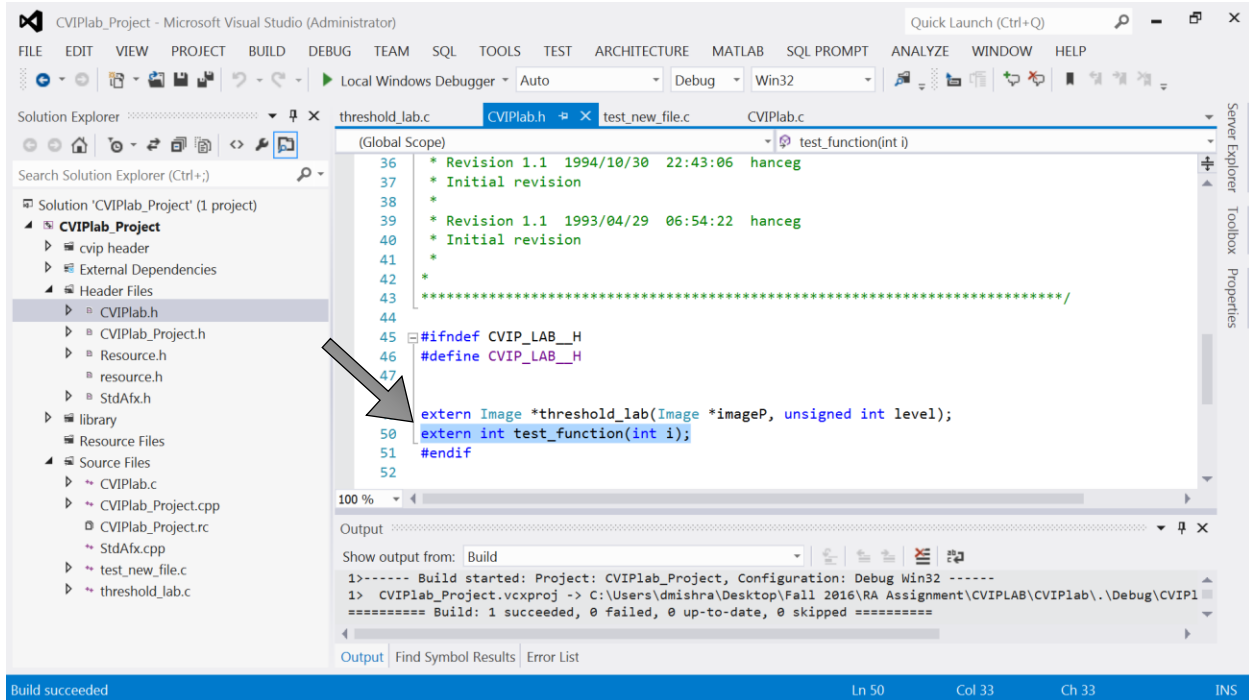
6. Double click on CVIPlab.h and CVIPlab.c to open them.

7. Find this line in the CVIPlab.h file:

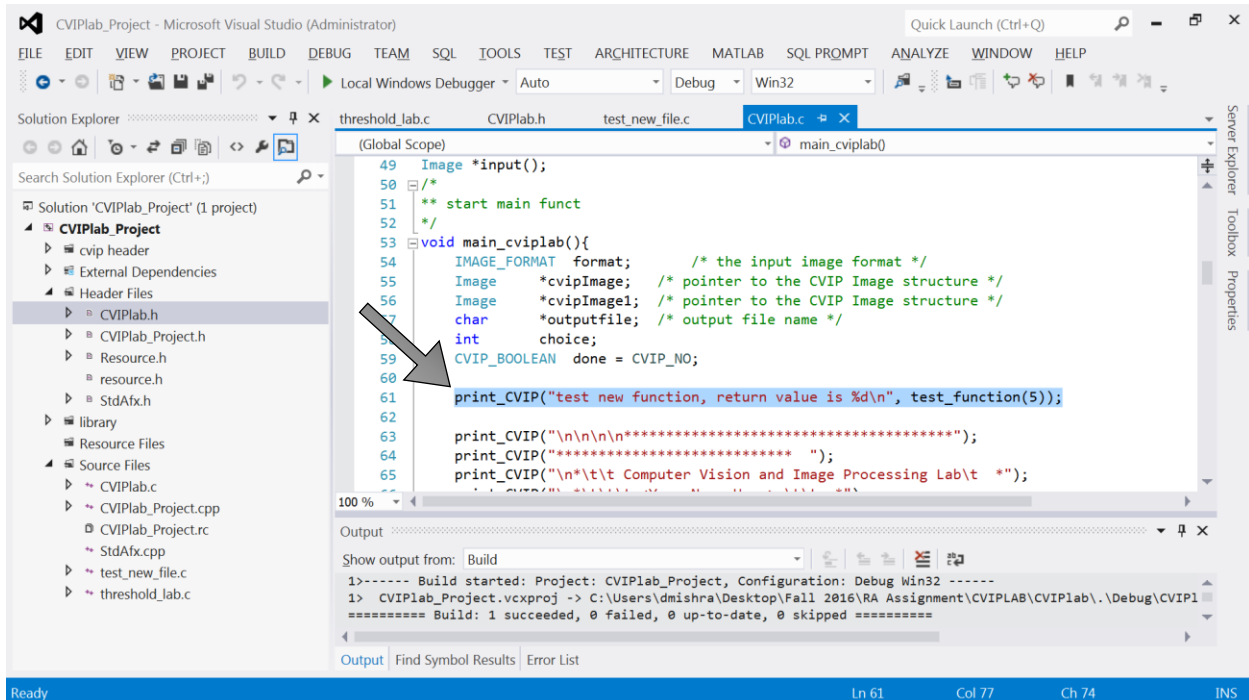
```
extern Image *threshold_lab(Image *imageP, unsigned int level)
```

And directly after it add a new line (see screenshots below):

```
extern int test_function(int i);
```



(a) Declare the new function in the header file CVIPlab.h



(b) Call the new function in CVIPlab.c.

8. Build the project again. It should pass the build without any error (Warning messages are OK).

9. Call this function in the main function of CVIPlab.c by inserting the following line:

```
print_CVIP("test new function, return value is %d\n", test_function(5));
```

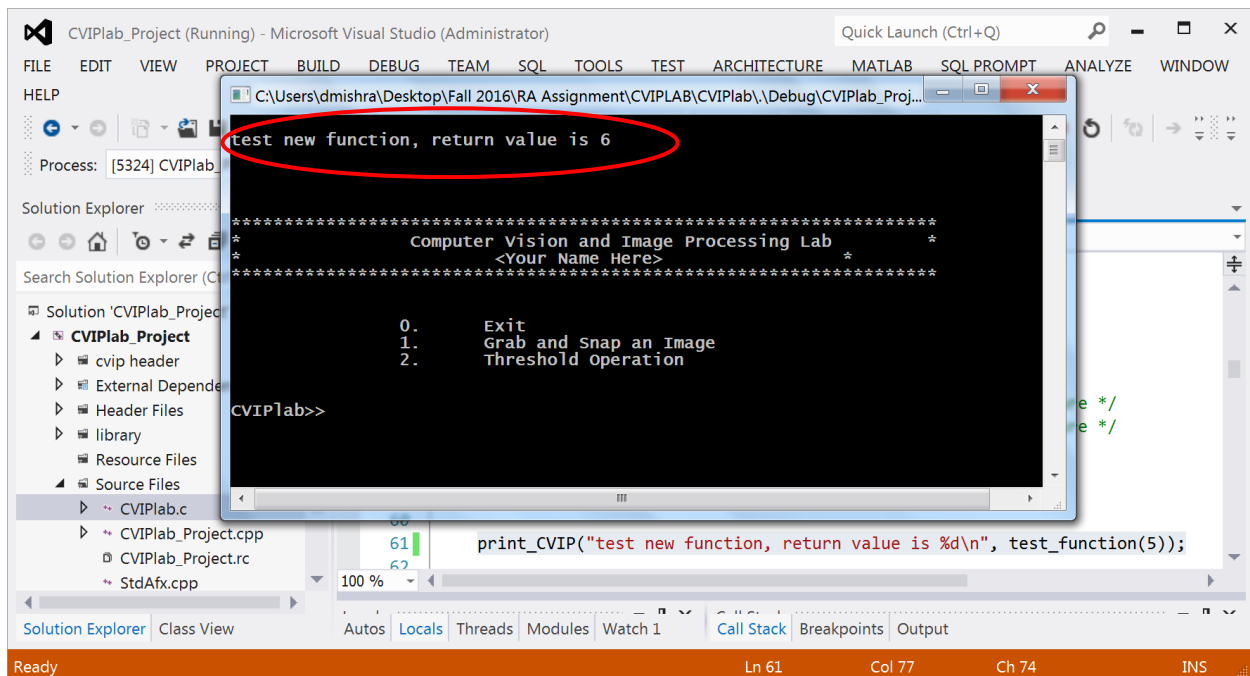
after the function declarations; as shown in (b) above.

10. Select *Build->Build Solution* to build the project. There should be no errors (again, you can ignore warnings)

11. Run the project by pressing F5. You should see:

```
test new function, return value is 6
```

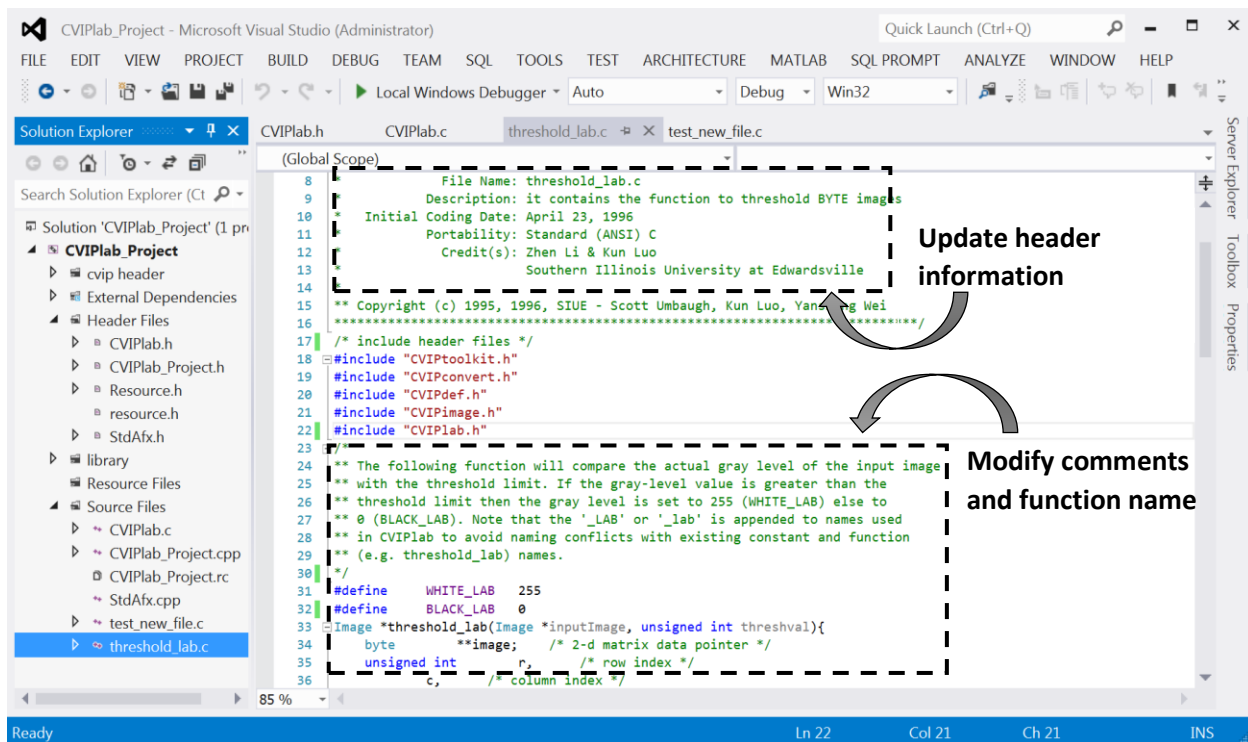
in the second line of the console window as shown below. If you see this, you have successfully added a new function to the CVIPlab project.



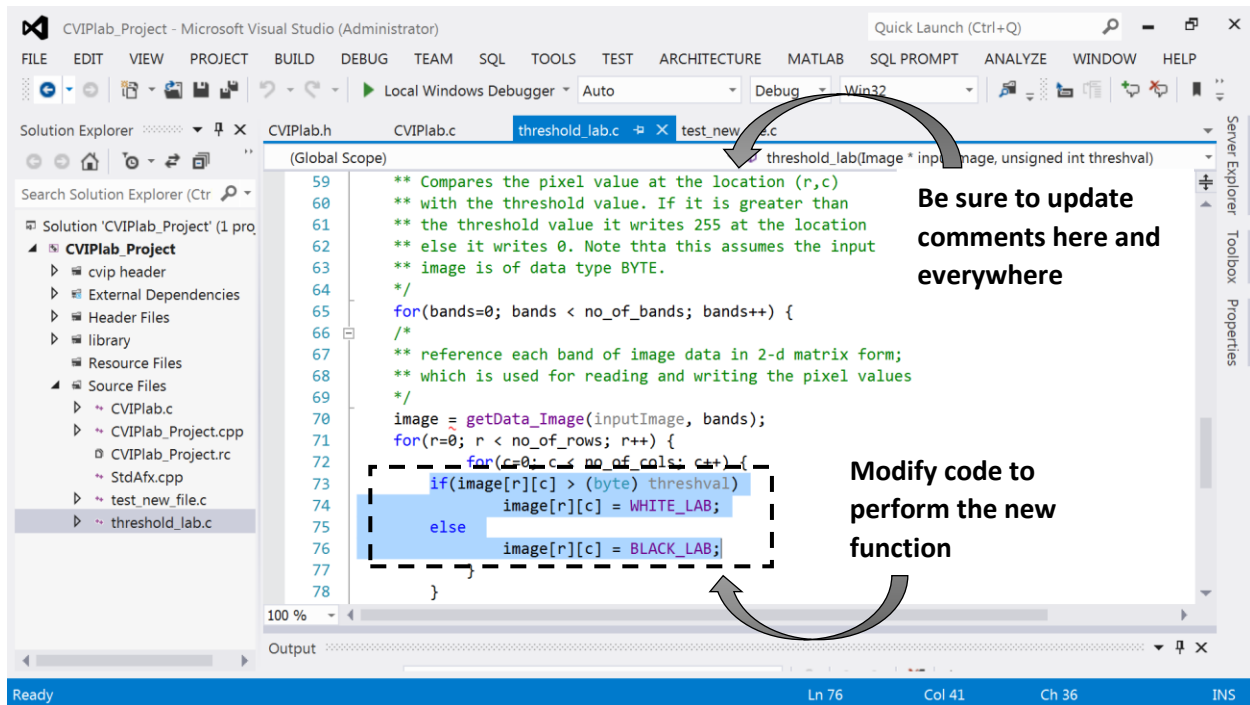
Using CVIPlab in the Programming Exercises

The previous section outlines the mechanics of adding a function with Visual C++. To follow the existing file organization and program format using any compiler, do the following:

1. Create a file similar to `threshold_lab.c` for the *new_function*. The easiest method is to select the `threshold.c` file and perform a *Save As* the `new_function.c`. Next, edit the header to change the file name, description, modify the date, add your name, and change the old comments and the function name. The last step is to modify the code inside the band, row and column *for* loop to perform the new function, as shown:



(a) Create your new function by using the threshold function as a prototype. Edit the header to change the file name, description, modify the date, add your name, and change the old comments and the function name.



(b) The last step is to modify the code inside the band, row and column *for* loop to perform the new function.

2. Add the new function to the CVIPlab menu as shown in (a) below. Next, add a case statement for the function as shown in (b) below. The case statement code for case 2 can be copied and used by modifying *threshold_Setup* to *new_function_Setup*.

3. Add the *new_function_Setup* to CVIPlab.c, similar to *threshold_Setup*.
4. Add the function prototype to the CVIPlab.h header file:

```
extern Image *new_function(new_function parameters...)
```