GENE 121 **Assignment 4** **Fall 2016**

This assignment will be **completed in pairs**. You have been automatically matched with a partner.

At the start of the lab session please sit at the robot kit number indicated on the partner list. Please do not alter the physical construction of the robot.

When you are finished, please do not dismantle the robot. We will collect them at the end and get them back in the containers.

In this assignment you will be using loops for repetition, performing file I/O, pass by value functions, and using RobotC to drive a robot.

Setup (Complete or verify that these steps have been completed)

1. Turn on the NXT by pushing the orange button.
2. Connect the NXT to your computer using the USB cable.
3. Start the RobotC software.

Running a Sample Program

1. Click on **File > New File** to open a new file.

2. Type or copy the following RobotC program (as given – yes, there are two errors).

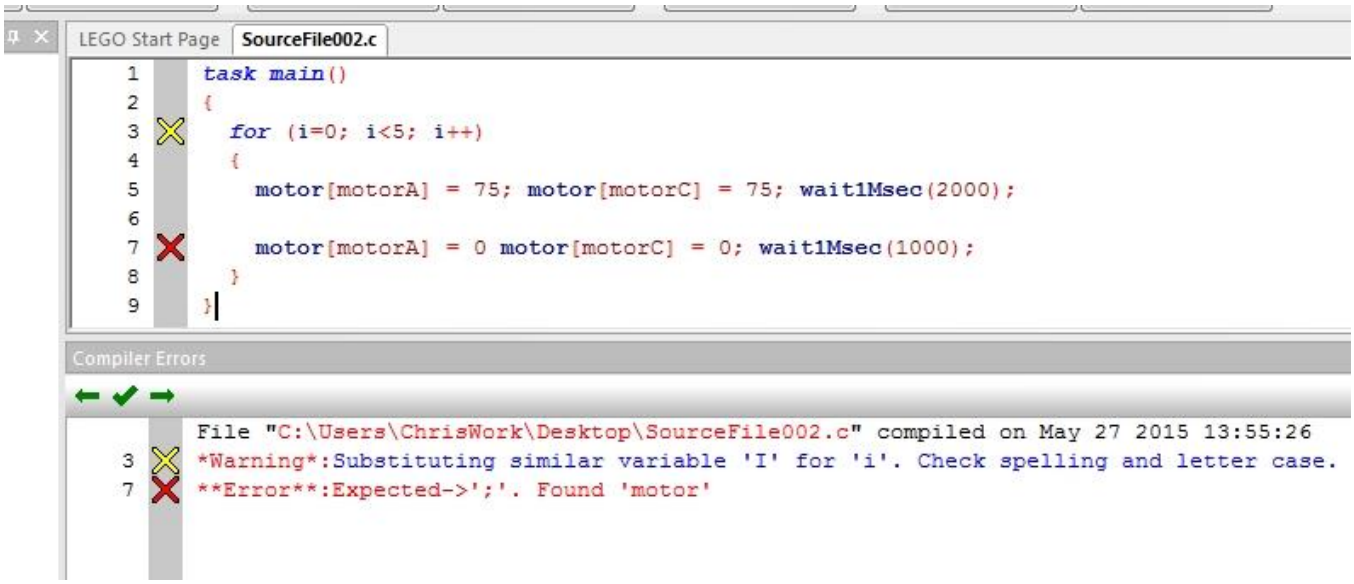
```
task main()
{
  for (i=0; i<5; i++)
  {
        motor[motorA]=75; motor[motorC]=75; wait1Msec(2000);

        motor[motorA]= 0 motor[motorC]=0; wait1Msec(1000);
  }
}
```

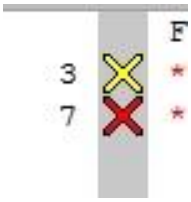
3. Save the program.

4. Compile your program.

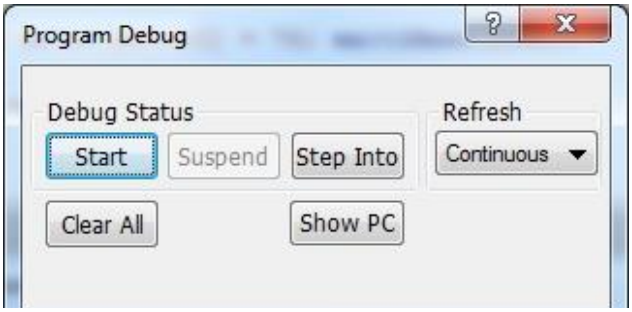
5. The compiler will indicate that it was unable to compile the file.



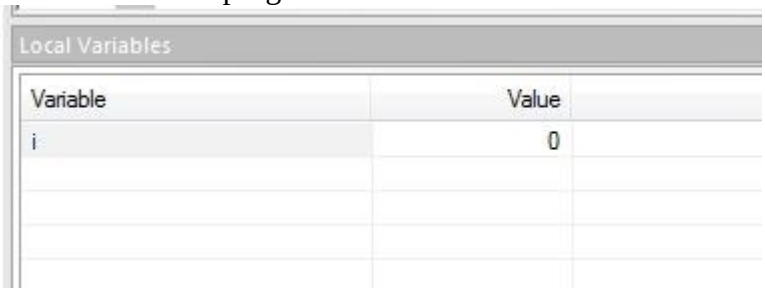
The error message window will describe the errors that are indicated with an X beside the line of code, as shown below. Fix the two errors and save the file.



6. Make sure your robot is on. Click on **Robot > Compile and Download Program** or **F5**. The program debug window will appear.



The compiler message window has been replaced by a debug window that will show your program variables as the program executes.



7. Hold your robot so that its wheels can spin freely, and click on **Start**. Verify that the robot operates as expected, and that the variable i counts up as expected.

8. Close the Program Debug window and disconnect the robot from the USB cable.

9. Push the orange button on the robot to select **My Files**, and then repeat to select **Software Files**. You should see the name of your program displayed. Press the orange button to select your program.

10. Put your robot on the ground and then push the orange button to run the program.

Question 1 (RobotC)

Draw a flowchart for a program that:

- Displays the day and your group number on line 0 of the display, e.g. `Tues 07`
- For 3 button presses **(frame of reference: stand behind robot)**:
  - If the right NXT button is pressed, the robot should rotate in place 90 degrees clockwise
  - If the left NXT button is pressed, the robot should rotate in place 90 degrees counter-clockwise
  - If the orange (centre) NXT button is pressed, the display should say "Allons-y!" and drive straight at top speed for 2 seconds.
- Finishes the program by doing something interesting with the robot

2. Write the program described above.

Demonstrate your working program to a TA, and then print and **submit your code with your flowchart** that you created in step 1.

Question 2 (RobotC)

For safety reasons, in this program, when the robot bumper hits something (touch sensor pressed) the robot should immediately stop moving and end the program.

a) Write a **well-named boolean** function which:
   - Receives a motor power and distance (measured from the front of the robot)
   - Drives forward at the given power until the front of the robot is at the given distance from an object
   - Turns 180 degrees
   - Stops driving
   - Returns true if the function ended because the robot read the given distance on the ultrasonic sensor, and false if the bumper was pressed.

b) Write a program which does the following:
   - Displays the day and your group number on line 0 of the display
   - Waits for the orange button to be pressed
   - Repeatedly calls your function from (a) to drive forward until the front bumper is pressed
     - Note: this may require that you change your function

Question 3

Chris' Message Delivery Service has set up a high-tech Message Delivery Catapult. He has built a file, **`message.txt`** with the following format:

- The first line of the file is the latitude degrees and minutes and the longitude degrees and minutes (all integers) that correspond to the GPS location of Chris' launch site
  - Note: There are 60 minutes per degree.
- The second line of the file is the number of messages to be sent
- The remainder of the file is a list of the latitude degrees, minutes and longitude degrees, minutes corresponding to all the delivery sites.

a) Write a function that receives integer values of degrees and minutes (for either latitude or longitude) and returns a double representing the location in decimal degrees. (**Ex: 60 degrees, 30 minutes = 60.5 degrees.)** Use the following function header:

```
double convDeg(int deg, int mins)
{
      //insert code here
}
```

b) Write a function that receives two doubles representing latitude and longitude in decimal degrees and returns the x coordinate. (1^o longitude ≈ 111 km x cos(latitude angle) km)
For example, 60.0 degrees latitude and 2.0 degrees longitude will return 111.0 (km).

c) Write a function that receives a double representing latitude in decimal degrees and returns the y coordinate. (1^o latitude ≈ 111 km)
For example, 1.5 degrees latitude will return 166.5 (km).

d) Write a main program which:

- Opens the **`message.txt`** file and verifies that it opened correctly
- Reads the location of Chris' launch site and converts it to Cartesian (x,y) coordinates using the functions you coded above
- Reads the number of delivery sites from the file, then
- For each delivery site in the file:
  - Reads in the location of the delivery site, converts to Cartesian coordinates and outputs the distance (in km) and angle (in degrees) from Chris' launch site to the delivery site.
    - You must use your functions above to handle the conversion

Submit your code with output and make sure to state your assumption as to what an angle of 0 corresponds to.