NAMES: _____

COMP256 – Computing Abstractions
Dickinson College
LAB #5
Compiled and Interpreted Languages

Introduction:

Human's generally prefer to write programs in high-level languages. However, computers can only fetch, decode and execute machine language instructions. We have discussed the two primary ways in which instructions in high-level language programs are converted to machine language so that they can be carried out by a computer: translation and interpretation.

In this lab you will gain some experience with and learn a little more about each of these approaches by working with a "Hello World" program in each. Note that "Hello World" is just a commonly used name for a simple first program in a new language[1]. These programs often just print the string "Hello World," thus the name.

Definitions:

1. Briefly describe how a source code program in a translated language is executed on the hardware of a physical computer.

2. Briefly describe how a source code program in an interpreted language is executed on the hardware of a physical computer.

[1] If you are curious you can check out this blog post on the history of the "Hello World" program: https://www.thesoftwareguild.com/blog/the-history-of-hello-world/

Preliminaries:

3. Open a Terminal window and create a directory named COMP256-Lab5. Give the command that you used.

4. Change into your COMP256-Lab5 directory. Give the command that you used.

5. List the files in your COMP256-Lab-5 directory (note: there won't be any, but you should confirm this). Give the command that you used.

A Compiled Language:

The short program below is a "Hello World" program in the compiled language C.

```
#include <stdio.h>
int main() {
   printf("Well hello world\n");
   printf("How you been\n");
   printf("Good to see you, my old friend\n");

   return 0;
}
```

6. Enter the above program into a text editor and save it in your lab directory with the name `HelloWorld.c`.

7. What files are in your lab directory now?

8. Because C is a compiled language the program must be compiled before it can be run. The lab machines all have a C compiler named `gcc` installed. Use the following command to compile your "Hello World" program:

```
gcc HelloWorld.c
```

This command says to run the compiler program (`gcc`) and use the source code file `HelloWorld.c` as input.

9. What new file now exists in your lab directory? This newly created file was generated by the C compiler and contains the executable machine language equivalent of your C language `HelloWorld.c` program.

10. To run this program use the command below. The `./` at the beginning of the command is Unix shorthand for "the current directory." So, the command below says "look for the program `a.out` in the current directory, and run it."

```
./a.out
```

What output is generated when you run the program?

11. Change the source code for your "Hello World" program to print "Well `hello Name1 and Name2`" where `Name1` and `Name2` are replaced with your names. Do not compile this program yet.

12. Run the `a.out` program again. Did the output change? Briefly explain why or why not.

13. Now compile your "Hello World" program and run `a.out` again. Did the output change? Briefly explain why or why not.

14. Briefly summarize the steps for making a change to the behavior of a program in a compiled (i.e. translated) high-level language.

15. Make the following changes to the source code for the "Hello World" program:

- Introduce some syntax errors by:
  - Removing the `;` at the end of the second `printf` statement.
  - Removing the `)` at the end of the third `printf` statement.
- Add another line that uses a `printf` to display "`Sometimes I feel as cold as steel`" at the end of the output.

16. Compile your modified "Hello World" program. How many errors are reported? What are the errors that are reported?

17. Run the `a.out` program again. Did the output change? Briefly explain why or why not.

18. Fix the errors you introduced above. Recompile and run the `a.out` program again. Did the output change? Briefly explain why or why not.

19. As an aside it is worth noting that there is nothing special about the name `a.out`. It is simply the default name that is given to the executable file generated by the compiler on Unix based operating systems (e.g. MacOS). Do some research on-line and figure out how to rename the file `a.out` to `HelloWorld` using the command line. What command did you use?

20. Run your renamed program. What command did you use?

An Interpreted Language:

The short code segment below is a web-page containing a "Hello World" program in the interpreted language JavaScript.

```
<HTML>
<BODY>

  <script type="text/javascript">
     document.write("Well hello world<br>");
     document.write("How you been<br>");
     document.write("Good to see you, my old friend<br>");
</script>

</BODY>
</HTML>
```

The interpreter for JavaScript is usually embedded inside of a web browser, and thus the source code for JavaScript programs often appears within a web page. The elements of the with < … > are called tags. The `<HTML><BODY>` tags define the start of the web page. The and `</BODY>` and `</HTML>` define the end of the page. The JavaScript source code for the "Hello World" program appears between the `<script>` tag and a `</script>` tag.

21. Enter the above web page and program into a text editor and save it in your lab directory with the name `HelloWorld.html`

22. To execute a JavaScript program within a web-page, the web page containing the JavaScript source code is loaded into a web browser. The browser then uses its JavaScript interpreter to execute the code between the `<script>` tags. Use the "File Open…" option on the "File" menu in your browser to open you `HelloWord.html` page. What output appears in the page when the browser runs the program?

23. Change the first line of output to be "`Well hello Name1 and Name2<br>`" where `Name1` and `Name2` are replaced with your names. Click the reload button in the browser. This causes the browser to read the source code for the page and thus the JavaScript program as well.

24. Based on what you have done so far, briefly summarize the steps for making a change to the behavior of a program in an interpreted high-level language and distinguish it from that for a translated (i.e. compiled) language.

25. Make the following changes to the source code for the "Hello World" program:

- Introduce some syntax errors by changing the last two `document.write` statements so that:
  - one says `document.right`
  - and the other says `document.left`
- Add another line that uses a `document.write` to display "`Sometimes I feel as cold as steel`" at the end of the output.

26. Click the reload button in the browser. How does the output change?

27. Based on what you just saw, when does an interpreted program a detect syntax error in the source code? How does that differ from when syntax errors are detected in a compiled program?

28. If we ask nicely the browser will show us what errors the JavaScript interpreter encountered. Under the "View" menu, find the "Developer" option and select "JavaScript Console." How many errors were detected? How does this differ from what happened with the compiled program? What do you think explains this difference?

29. Correct the errors in the source code for your "Hello World" JavaScript program and run it to ensure that you have them fixed.

Got all That?

Based on the previous exercises you should now be able to predict what will happen in the following scenarios. Make your predictions before testing them as a way to check your understanding of the differences between compiled and interpreted languages.

30. Add the code snippet below to the source code for your C language "Hello World" program. Place this code above the `printf` statements but inside the `main` function. **Do not compile the program yet.**

```
int a = 7;
if (a == 2) {
   printf("awww boogers";
}
```

31. Will the body of the above `if` statement ever execute (i.e. is the condition of the `if` statement ever `true`)? Why or why not?

32. The code in the body of the if statement contains a syntax error. What is the error?

33. Do you think this error will be reported if you compile the program? Briefly explain why you think that.

34. Now compile the program. Did it behave as you expected? If not, briefly explain what you had you misunderstood and why it behaves as it does.

35. Add the code snippet below to the source code for your JavaScript language "Hello World" program. Place this code above the `document.write` statements but inside the `<script>` tags. **Do not reload the program yet.**

```
a = 7
if (a == 2) {
   document.left("awww boogers");
}
```

36. Will the body of this `if` statement ever execute?

37. The code in the body of the if statement contains a syntax error. What is the error?

38. Do you think this error will be reported if you run the program? Briefly explain why you think that.

39. Now run the program by reloading the page in the browser. Did it behave as you expected? If not, briefly explain what you had you misunderstood and why it behaves as it does.

Working in Compiled and Interpreted Languages:

If you have completed all of the above during the lab period, continue to work on the following exercises. If you did not complete all of the above during the lab period, these exercises are not required.

40. Modify your C language "Hello World" program so that it prompts for the name of the person to say hello to, reads a name from the keyboard and then prints something like "`Hello Name!`", where `Name` is replaced by whatever the user entered.

I expect that most of you do not know C, so you will need to do some research on-line to figure out how to read input from the user in C and then how to print it. I suggest the following strategies:

- Search around for examples that look familiar and do things close to what you want to do. Find some that use a function to read input from the user. Find some that use `printf` to display input read from the user. Then try to generalize and adapt these to your task.
- Don't just go with the first thing you find. Look at a number of examples before trying anything. Find examples that you have some intuition about. Yea you might not know C, but you know how to program. So you will find that you have intuition about what things do. For example, you know about arrays in Java, and arrays in C aren't all that different.
- Avoid the trap of moving toward things that are overly complicated and about which you have no intuition about what they are doing. Try a lot of simple things that should work before giving up on them. Often there is just one little thing that is preventing it from working. If you become sure the simple stuff can't work then you'll need to learn more – just plugging in really complex code is a recipe for disaster. NOTE: In this lab, the simple stuff will work!

Be sure to compile and run your program to ensure that it works. You might have to try several things and recompile multiple times but that's okay, you are learning.

41. Modify your JavaScript language "Hello World" program so that it prompts for the name of the person to say hello to, reads a name from the keyboard and then displays something like "`Hello Name!`", where `Name` is replaced by whatever the user entered.

The same strategies used to modify the C language "Hello World" program will work here.

Be sure to reload your program to run it and ensure that it works.

Submit the Code:

42. Compress your COMP256-Lab5 directory to a zip file and submit it to the Lab5 assignment on the course Moodle. One submission per lab pair will be sufficient.