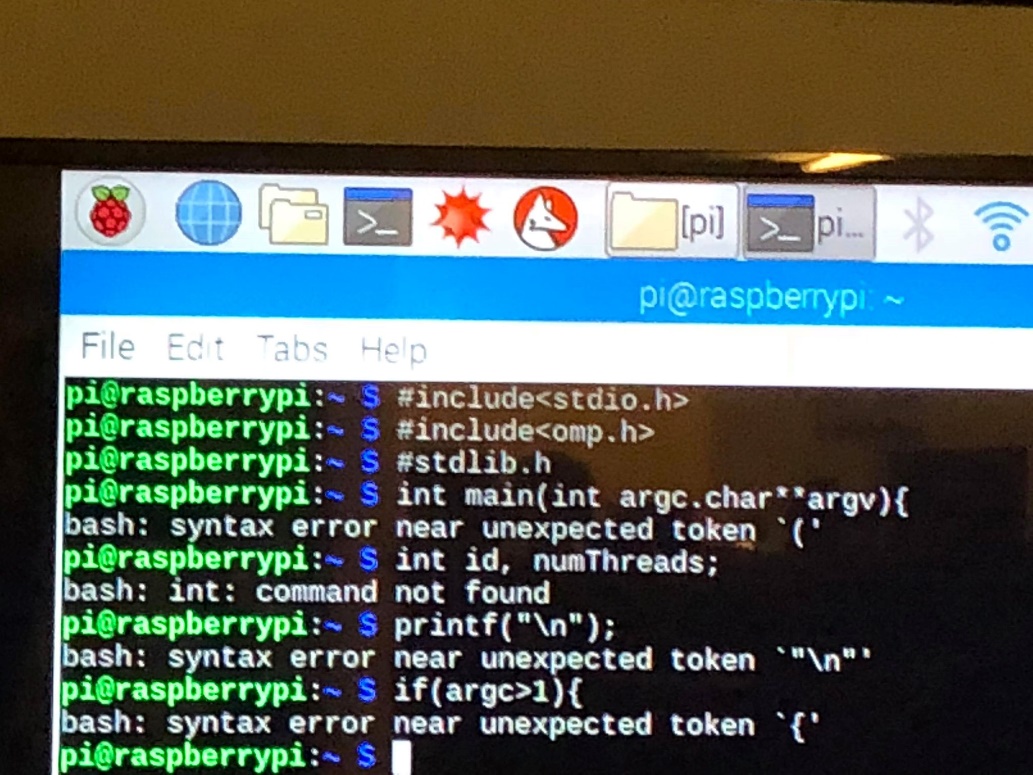
Developing Soft and Parallel Programming Skills Using Project – Based Learning

Fall – 2018

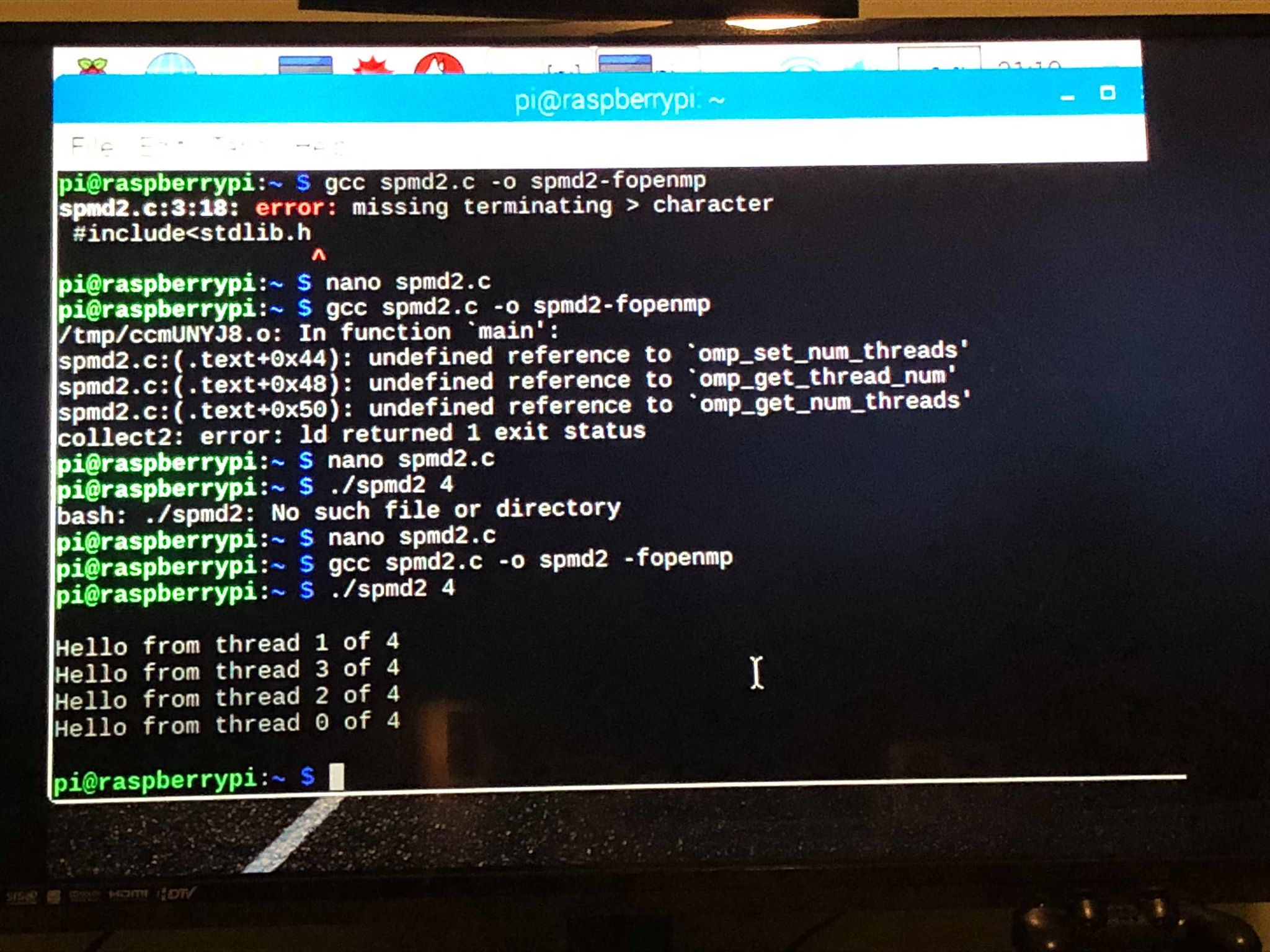
Group Name: Dream Team

Group Members: Michael Sawyerr, Victor Barron,  Basuamlk Woldatsadik, Amber Choi

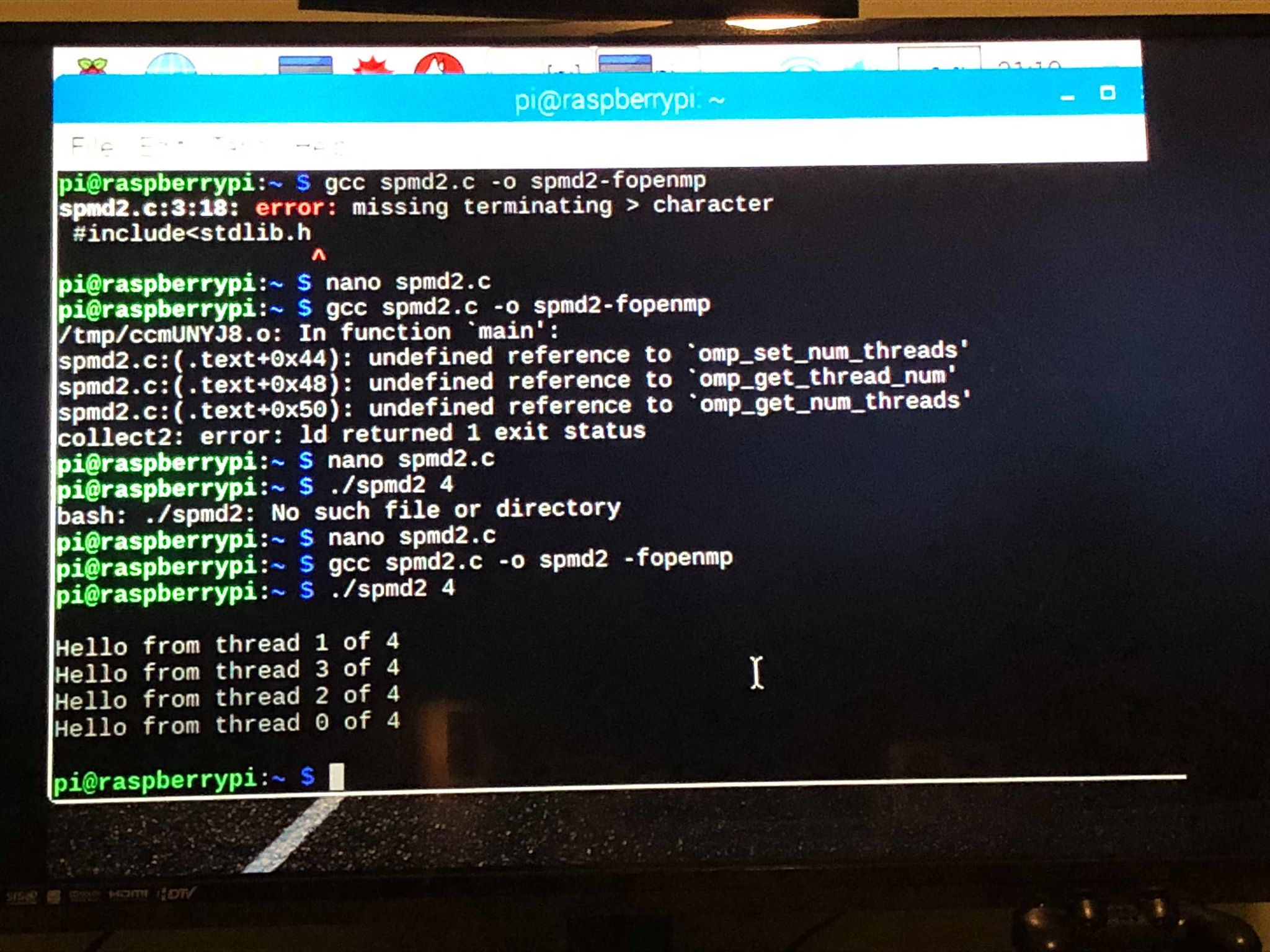
**Dream Team Raspberry Pi Report**  
  
What I have done was start understanding how the Raspberry PI works. Unlike some programming applications, you cannot run a program from the start, you would need to use the Editor known as “Nano.” Something I tested out during the task from the start was “What would Happen if I began typing in the code outside the Editor.”



Uninformed about my current situation, I found it interesting that I would constantly get an error after every inputted line. This was until I thoroughly read through and understood the editor, “nano spmd2.c”. Upon entering the code using the editor I was met with 2 errors at first, one being a missing terminating character,

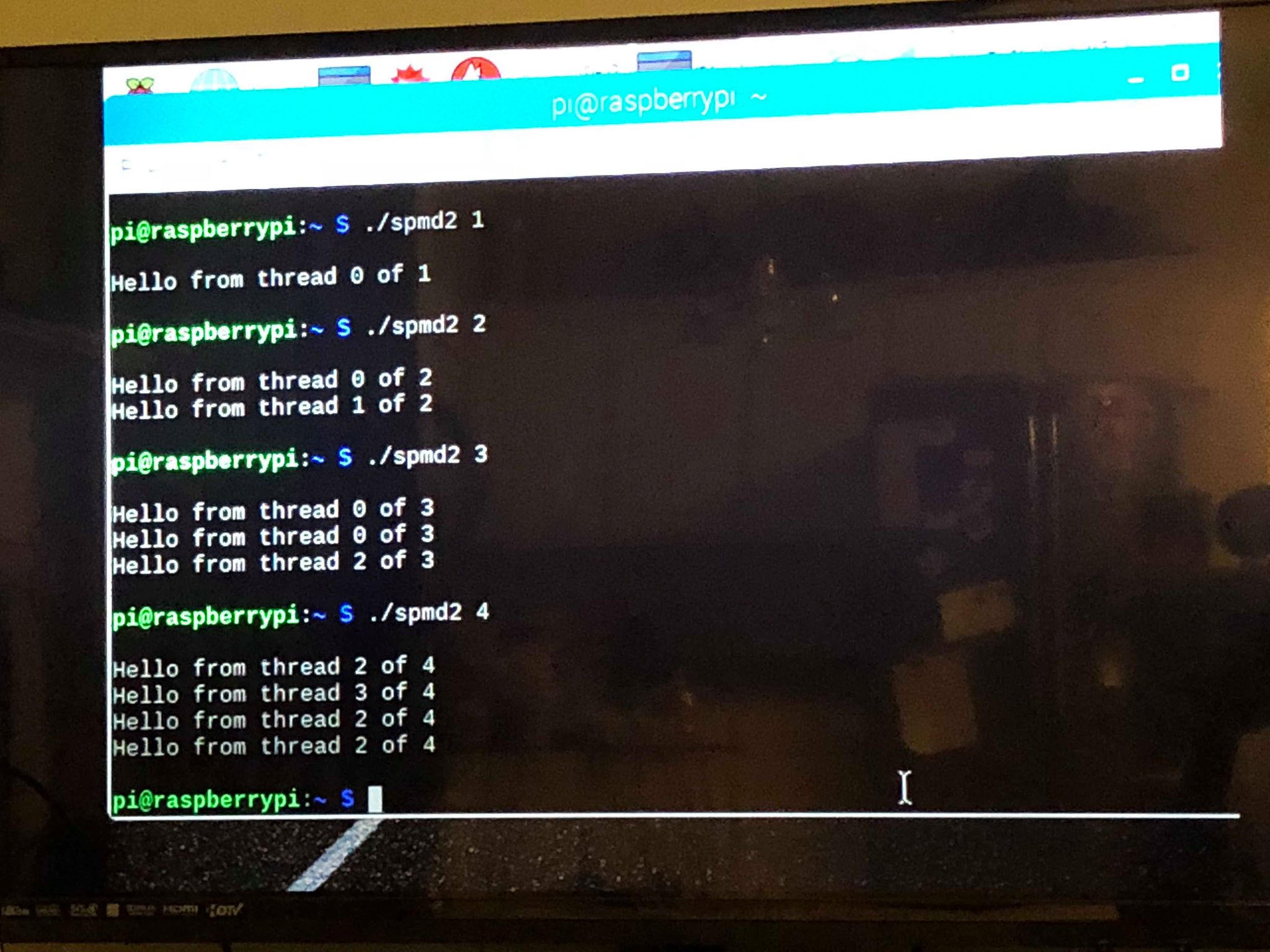


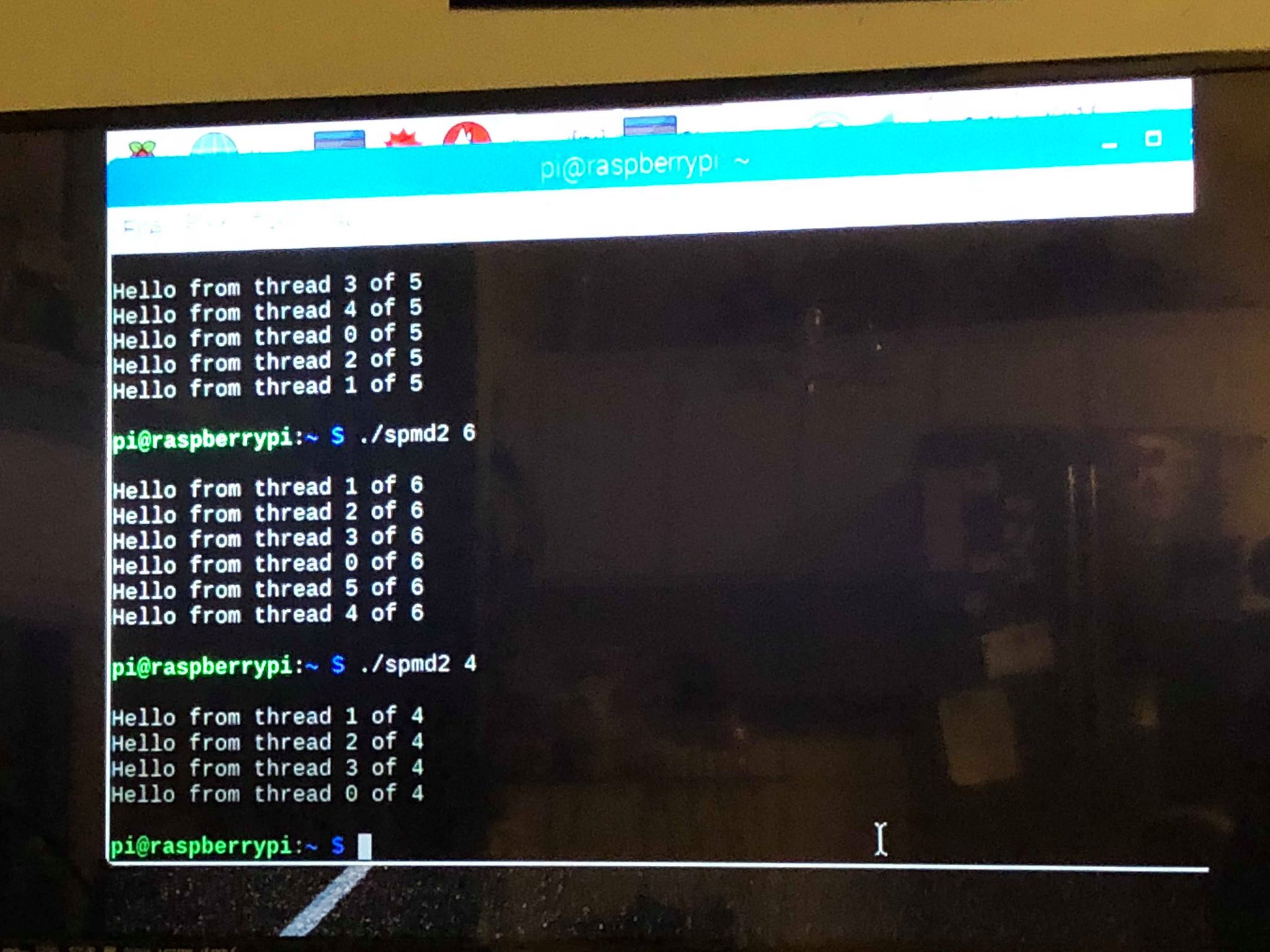
And undefined references.



While the solution to the terminating character was as simple as adding the “>” character, I had to progress further through the task to figure out the undefined references error. What I particularly found interesting was after applying the steps in section 2.4.1, “ //int id, NumThreads;” running the program, and then removing the “//,” the error ceased and the program continued running regularly. Moving onto running the program itself, I observed the code segment in charge of making the executable program, “ gcc spmd2.c -o spmd2 -fopenmp” as well as the code segment in charge of running the program, “ ./spmd2 4.” If the code segment has any lingering errors, the executable program will not be created, and therefore cannot run.

After running the code several times with 4 threads as instructed, as well as running it using a different number of threads. I noticed that while order was not only the issue, but a thread number appears more than once, rather than having its own number.



This was due to the variable being declared outside the block, causing all the threads to share the same memory and therefore ID. The tasked then showed me how to fix this issue, by adding a “//” in front of “id” and “numThreads” as well as adding an “int” in front of the id and numThreads variables. This was the result. I also used multiple numbers 4 and up for the threads. 

I thought it was interesting how it still wasn’t in order, however how C is similar to Java, I assumed there would need to be a loop to check the order.

* **Planning and Scheduling:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Names | Email | Task | Duration (Hours) | Dependency | Due Date | Note |
| Michael Sawyerr | msawyerr2@student.gsu.edu | -Invite TA to Slack (?)  -Set up raspberry pi  -work on parallel programming basics |  |  |  |  |
| Victor Barron  (Coordinator) | vbarron1@student.gsu.edu | -Set up raspberry pi  -work on parallel programming basics  -finish second half of report  -proofread report |  |  |  |  |
| Basuamlk Woldatsadik | bwoldatsadik1@student.gsu.edu | - assist in setting up raspberry pi  -parallel programming skills: Foundation questions |  |  | - |  |
| Amber Choi | achoi11@student.gsu.edu | -create Github  -parallel programming skills: Foundation questions  -youtube video  -set up report |  |  |  |  |

Github = dreamteam3210

Password = Teamwork3210

* **Parallel Programming Skills: Foundation**

**(5p) Identifying the components on the Raspberry PI B+ .**

The components are CPU, RAM, USB (2), ethernet controller, the ethernet, power (2), camera, HDMI, and display.

**(5p) How many cores does the Raspberry Pi’s B+ CPU have?**

Four (Quad Core)

**-(8p) List four main differences between x86 (CISC) and ARM Raspberry PI (RISC).**

The most prominent main difference is their instruction sets. CISC has a larger, complex instruction set with more features, allowing for more operations and addressing modes.

ARM offers more general purpose registers than CISC, and operates only on registers.

Most of ARM’s instructions allow for conditional execution.

X86 uses little-endian format, while ARM changed to BI-endian and allows for the endian to be switched after version 3.

**-(6p) What is the difference between sequential and parallel computation and identify the practical significance of each?**

Parallel computation is the simultaneous use of multiple compute resources to solve computer problems. These problems are broken down into many parts with its own instructions for each, thaat execute simultaneously in different processors.

Sequential computation is when the problems are broken into many sets of instructions and executed sequentially, one at a time, on a single processor.

**-(5p) Identify the basic form of data and task parallelism in computational problems.**

Data parallelism is when the same computation can be applied to multiple data items. The parallelism is proportional to input size.

Task parallelism is when the parallelism is organized functions and their performance, and not around the data.

**-(6p) Explain the differences between processes and threads.**

Processes are the abstraction of a running program, and a thread is a lighter process that allows a process to be broken down into more smaller parts.

The memory is shared into a common memory in threads, but not shared in processes.

**-(3p) What is OpenMP and what is OpenMP pragmas?**

OpenMP became an industry standard since the 1990s, and receives native support with GCC compilers.

OpenMP pragmas are compiler directives that enable the compiler to generate threaded code.\* copy pasted lol pls help change format

**-(4p) What applications benefit from multi-core (list four)?**

The applications are database servers, compilers, multimedia applications, scientific applications (CAD/CAM), and overall any applications with thread-level parallelism.

**-(4p) Why Multicore? (why not single core, list four)?**

Multicore has gained more popularity over the years as many more applications are multithreaded and the nature of computer architecture is shifting towards parallelism. Single core on the other hand, has a more difficult time to make its clock frequencies higher than multicore, and has deeper pipelined circuits, which causes heat and speed of light problems, larger design teams, and a more difficult design.

* **Appendix:**

[www.github.com/dreamteam3210](http://www.github.com/dreamteam3210)

<https://dreamteam.slack.com/>

(Video Link here)