Riley Porter (rileymp2) and Clara (claram2)

**Vision**

On a high level, our project is a C Visualization Tool for learning and debugging small C programs. This is assuming that the C program already compiles, since we wouldn't have time to implement a compiler and there are already a bunch of great C compilers out there. This project is for students who are new to C and those who want to see state as it's updated. This is also for beginners that aren't used to the complicated pointers that C (and other languages) use. Hopefully this will help people understand the concepts of C more easily and make it a more friendly language.

A lot of people are visual and looking at code can be confusing at first. This project would bridge the gap between reading textual code and picturing what's happening. We want to display both pointers and the current state of variables so that it is really easy to step through code and find out where something is going wrong. This can be especially difficult with arrays or linked lists in C, so that would be somewhere we want to start with this project.

The current available alternatives aren't easy to find online, nor are they up to date. There is the gdb debugger, but that isn't intuitive for beginners and doesn't have the visual component. This is a compelling project and it is worth developing because computer science is growing and there will be more people wanting to study such complicated topics. There are lots of debugging tools out there and there are many libraries to add onto a debugger to make it more useful, but our novel approach is adding an easy to use visual aspect to C code.

Our top level goals are to make C easier to understand, learn, and visualize. The target customers would be instructors for courses for beginning C programmers. Universities or other learning institutions might also buy this for their students to enhance their learning. The scope of the product is for smaller C programs. Or focusing on a small section of a larger C program and only displaying a portion of the variables or code that is necessary.

**Software Architecture**

We will probably use javascript or a similar language to display the visual component. To parse the input code we will use regex and string manipulation to create a structure that models the code with variables, objects, dependencies, etc. We can build this in a piecewise manner, and implement the project for smaller pieces and build it up. This way, we won't run the risk of biting off more than we can chew. We can start with visualizing arrays or linked lists, individual functions, or certain types of variables and then go from there if we have time.

The system would comprise of a front page with a text box to enter C code and from there it would go to a second page where a user could select what parts of the code they are interested in. Finally, there would be a third page to step through the code which would not allow modifying the code after this point, and the page would display what is happening with the variables the user has selected.

We would implement this functionality with javascript and some libraries for the

visualization, regex and string manipulation to parse and modify the database, and json objects (or XML) and a dependency tree to store the current C code variables and functions.

From a technical point of view, this project is interesting because it involves parsing plain text and interpreting it as having meaning as a C program. This will involve complicated regular expressions and taking into consideration different styles of formatting code. Also, this will require a lot of understanding of how C works and the different pitfalls beginners will encounter.

**Challenges and Risks**

The biggest challenge to this project is that it could easily get out of hand as far as scope of how much to tackle. We could easily start this project with the goal of being able to visually depict multiple C functions, show many data structures, and also maintain state and pointers in our visualization. 8 weeks isn't that much time though, so implementing all of that could be difficult. To mitigate that, we intend to start small and build up the functionality slowly. First making sure everything works great for only displaying variables, and then moving on to more complicated topics like data structures, pointers, and functions. Hopefully we are able to accomplish all of the functionality that we want, but we recognize that having a fully functional smaller project would be more useful than a partially-functional larger project.

The other risk is that not everyone who works on this project might be familiar with C. This project will require some in depth analysis of turning plain text into meaningful C code, and that will be an interesting part of the project to tackle. With 6 to 8 people however, there is no reason why we can't get at least basic functionality working so that pointers and variables are displayed clearly for a user.