Figure 6.1: We do not quite live in a matrix, even though we programmed one in assignment 5. However, with our tools we have learned to program with this semester, you are now equipped to build some cool software!

6.1 Description

We have learned a great deal in this class! Part of becoming a good computer scientist, is well-programming! The best way to get better, is to build actual software that you can use. Dream up what software should exist in the world in
some domain you are interested in working in. And as you build that software, you will encounter various challenges that push you to learn more.

For this final assignment, you get to build what you proposed in assignment 5! The goal is this assignment is something you would carry on beyond this course as well! There is room for creativity within this assignment!

6.2 Task

The tasks for this assignment are the following:

1. Documentation:
   (a) Prepare a .pdf of word(.docx) file.
   (b) Name of the project (give it a descriptive and cool title)
   (c) A 3-5 sentence of description of what you accomplished.
   (d) A diagram of the classes and diagrams you created outlining the structure of the program.
   (e) A bulleted list of features to make the minimum viable program(MVP) (i.e. The features you completed).
   (f) A brief users guide on how to compile, run, and use the software. This should include a sample use case (a step by step example of what your problem your program solves)
   (g) An analysis of the outcome of your program. Did you achieve your proposed minimum viable product? If not why not?
   (h) A reflection on your design. What was the best or worst design decision you made. Use as much or as little details as you need, such that if you looked back on this project in 6 months you would agree with your rationale.
   (i) Test and Bug Report: Are there any known errors with this program? What inputs will cause errors. How did you test your program such that it was correct? List these items in a bulleted list or in a table.
   (j) A bulleted list of each of the .cpp and .h files and a once sentence description of what their main functionality is.
   (k) Include at least 1 screenshot of your program running.

2. Project Code
   (a) You will submit all of your code
   (b) It should be well-documented.
   (c) Your project should compile.
   (d) If you have any additional dependencies (external libraries you found for example), you should include how to obtain them in your README
6.3 Input Files

None given–you create them all from scratch this time!

6.4 Files

You will create the files all from scratch for the project.

You will have at a minimum the following files named exactly (including the exact capitalization):

- project.pdf or project.docx
- main.cpp
- README

6.5 Submission

You must test your program on the machines in Halligan (either physically or by ssh’ing into the homework.cs.tufts.edu server).

```bash
provide comp11 ps6 your_cpp_files README project.pdf
```

Listing 6.1: Submit Assignment

6.6 Expected Output

Your provided code samples should compile and run on the Halligan Machines. There should not be memory leaks!

(For this project, you should capture a screenshot somewhere in your project of your project working with the expected output.)

6.7 Evaluation/Rubric

- Make sure you test your program on the halligan computers (either physically or through ssh).
- Files are properly named as specified below.
- You include a file called README describing your assignment.
- Part of your assignment will be graded based on code style.
- The remaining part of your assignment will be graded based on the correctness, and that it works to the specification.
– You project should include all of the items in the tasks at a minimum (You may add more if you like).
– Your project document should be spell checked.
– Your project document should be complete, such that if there was a homework 7 other students would understand what you did and they could implement the bulk of the project.

6.7.1 Style

Use good style in your projects, and liberally use comments! Some examples are provided below that you may use as a template.

```cpp
// if-statements
if (a > b)
    // Notice how the indentation nicely lines up with the conditionals

// for-loops
for (char c : s)
    // Range-based for loops are useful if we are not modifying the value.

int myArray[10];
for (int i = 0; i < 10; ++i)
    // If we are modifying values, then use a regular for-loop
    myArray[i] = i;

// structs
struct node
{
    node* next;
    int data;
};

// classes
class myClass
{
public:
    myClass();
    ~myClass();
    myClass();
};
```

Listing 6.2: Some Good Style Examples