Tradd Schmidt

Dr. Scott Hegen

CSC 226

April 8, 2017

CSC 226 Final Project: Software Design Document

**My Proposed Project Extends the A8: Error Detection Assignment**

By Tradd Schmidt

**MOTIVATION:** I have always been slightly curious as to how computers interpret 1’s and 0’s and represent them into characters. This will be an interesting opportunity to work with ASCII characters and understand them better. It will also be a useful program for real world application.

**PURPOSE:** The primary purpose of this program is to add parity bits to a 7-bit string of ASCII character codes to prevent errors in transference of data.

**RESOURCES:** This program is based on the A8: Error Detection Assignment. I will be extending this assignment by adding in the following functionality.

**INITIAL DESIGN PLAN:**

I will complete this assignment by adding the following functions:

The original implementation of A8 included the following functions:

* test\_it():
* parity\_test\_suite():
* is\_binary():
* is\_div\_by\_sevens():
* split\_into\_sevens():

My implementation for the final project will include:

* Refactoring the code to use classes. The CRC card for the class structure is included.
* Implementing the functionality to take a string of characters, convert them to binary, and return the binary pieces with the parity bit added as a list which will be a function called string\_parity()

**CRC CARD:**

|  |  |
| --- | --- |
| **Class name:** |  |
| **Class Methods:** | **Class Collaborations (other classes):** |
| * \_\_init\_\_(): initializer which takes an input of either binary numbers, a string of characters, or a file. * \_\_str\_\_(): Allows the string of text or binary to be returned depending on which is being used. * is\_binary(): Checks to see if the input into the function is binary. * is\_div\_by\_sevens(): Checks to see if it is divisible by seven * split\_into\_sevens(): If is\_div\_by\_sevens() passes then it splits the characters up * convert\_to\_ascii(): Converts to ascii if needed and returns it with the parity bit added. * extract\_text(): takes all of the text out of a string and puts it in a list to be used in convert\_to\_ascii(). |  |
| **Class Data:** | **Class Collaborations (other classes):** |
| * self.text #instance variable which holds the text that will be used. * self.binary #instance variable which is used to indicate if self.text is binary or not. |  |

**SUMMARY:**

I had to adjust my design a bit for the program to work. I was originally going to include the functionality of being able to convert a file of text to the ASCII code, but I could not get it to work no matter how hard I tried. In addition I renamed some functions and changed some class data and have updated the CRC Card appropriately. My final result is a program that can take all characters or binary numbers and convert them to ASCII if need and return them with the parity bit added on. It took me about 5 hours to complete the coding portion. The dictionary that was used took quite a while to create and troubleshooting took some time.

The hardest challenges to overcome was representing \, ‘, and “ in the dictionary. After some trial and error I figured out to represent \ you have to write two of them right next to each other like \\. ‘ and “ were also tricky because everything in this dictionary is represented as strings so I needed to wrap “ in double quotes like “””. However, python will just consider this a triple quote. In order to fix this I had to write these to parts as “’” which is an apostrophe between two quotation marks and ‘”’ which is one quotation mark between two apostrophes. These three problems can be found in lines 95, 96, and 105.

**FILES:**

* Final\_Project.py
* Final\_Project\_Test\_Suite.py

**VIDEO:** [**https://www.youtube.com/watch?v=kqGglrDa124**](https://www.youtube.com/watch?v=kqGglrDa124)

**INSTRUCTIONS:**

1. Open cmd
2. Type “cd “(the address of where the file is located)””
3. Type “python “(the name of the file to be run)”
4. Type what you want the parity bit added to
5. If it is binary code type True. If not type False.
6. You now have your ASCII binary with the parity bit added.

**ERRORS:** There are only 2 errors that I know of. If text is copied and pasted to be used in the program, and the text has enters being used, the new line that was created by pressing enter is not represented as \n but actually is used as if you pressed enter into the program. In addition, the binary portion of the code can only take 7 numbers at a time. Using any different number causes errors.

**REFLECTION:** I think this assignment is a great chance to show people the value in writing reusable code. Hopefully, it also shows the importance of test suites. In addition it allows people to get creative in how they can expand previous thinking. This also shows them that just because you finish one program does not mean you will never have to adjust the code or add to it.

**REFRENCES:** Dr. Scott Heggen helped me with my test suite. I was having errors with it running because I had to things in my code named the same. He pointed this out to me and it fixed my code.