1 Objectives

The objectives of this project are to gain experience:

1. Working with the basics of the C language.
2. Working with strings and loops.
3. Converting strings to numbers using the `strtol` library function.
4. Working with a Makefile.

2 ASCII Background

A standardized model for representing characters on a computer is ASCII (American Standard Code for Information Interchange), where each character is encoded as an integer. For example, the following three characters of the English alphabet are represented in ASCII with their associated decimal values: a = 97, A = 65, (= 40.
Here is a table of ASCII encodings: https://www.rapidtables.com/code/text/ascii-table.html

3 Project Requirements

Big picture: You will write a program called `asciitotal` that takes two strings as inputs (i.e., passed via `argv`), where the first input can be any string, and the second input represents a whole (integer) number. The program first computes and displays the total of the ASCII values of each character in the first input string. It then multiplies that total ASCII value by the second input number and displays the result. For example, a program run might look like the following:

```
$ ./asciitotal hello 3
The ASCII total of the string = 532
The total * multiplier        = 1596
$
```

The total of "hello" is 532 because it consists of the ASCII encodings of 104 + 101 + 108 + 108 + 111.

3.1 Program Requirements

1. The program source code shall be written in a file called `asciitotal.c`.
2. The program `asciitotal` shall take two strings as input, as shown in example above. The first input is a string of any length and content, while the second input string is expected to represent a decimal integer between 1 and 100, inclusive.
3. The program `asciitotal` shall detect when the input is invalid. When `asciitotal` detects invalid input, it shall display a meaningful error message to the user, and then immediately exit with a value of EXIT_FAILURE (see "man 3 exit").
   Here are a few ways that input might be invalid:
   a. Too many or too few arguments were provided on the command-line.
   b. The second input does not represent a whole number, e.g., 1.1, abc, or 1ab.
   c. The second input represents a number less than 1 or greater than 100.

4. After verifying that the input is valid, the program shall:
    a. Display the total value of the ASCII encodings of the input string as shown above.
    b. On the next line, display the total ASCII value multiplied by the multiplier as shown above.
    c. The "=" characters of the output lines shall be aligned as shown above.
    d. The program shall return EXIT_SUCCESS (assuming no errors).
5. No global variables may be used in your program.

3.2 Makefile Requirements

You will also write a Makefile that will be able to create the `asciitotal` program. Specifically, the Makefile shall have the following targets only:

1. `all`
   When "make" or "make all" is entered on the command line, it **only** tries to create the `asciitotal` program. If `asciitotal` is newer than `asciitotal.c`, then `make` should <u>not</u> compile `asciitotal` again.
2. `asciitotal`
   When "make asciitotal" is entered on the command-line, this target will create the `asciitotal` program **only** if `asciitotal.c` is newer than `asciitotal`.
3. `asciitotal.tar`
   When "make asciitotal.tar" is entered on the command-line, this target shall create `asciitotal.tar`, which will contain the final versions of your "Makefile" and "asciitotal.c". The command for creating this tar file is:
   `tar -cvf asciitotal.tar Makefile asciitotal.c`

4 Submission

Post your final `asciitotal.tar` file to Sakai before the assignment deadline. **<u>Reminder that late submissions will not be accepted</u>**. If you are not finished by the assignment deadline, then turn in what you have completed to get partial credit (rather than getting a zero for turning in nothing).

5 Grading

Grading will be based on the guidelines below. Partial credit will be assigned for deviations from successfully completing any of the below areas, and I reserve the right to include other reasons for deductions.

1. (10) A proper working Makefile was provided, as specified in Section 3.2.
2. (20) The `asciitotal.c` code compiles without errors when the `gcc` compiler is used.
3. (10) The `asciitotal.c` code compiles without warnings when the `-Wall` compiler option is used with the `gcc` compiler.

4. (10) No segmentation faults or other crashes occur when the program is run.
5. (20) The program handles *valid* input as specified in Section 3.1. In particular, the following tests for valid input will be performed:
   a. `./asciitotal hello 1`
   b. `./asciitotal world 10`
   c. `./asciitotal W 100`
6. (20) The program handles *invalid* inputs as specified in Section 3.1. In particular, the following tests for invalid input will be performed:
   a. `./asciitotal`
   b. `./asciitotal hello`
   c. `./asciitotal hello world`
   d. `./asciitotal hello world 1`
   e. `./asciitotal Jan 3rd`
   f. `./asciitotal hello 0`
   g. `./asciitotal hello 101`
   h. `./asciitotal hello 1.1`
   i. `./asciitotal hello 999999999999999999999999`
7. (5) The `strtol` library function was properly used to convert the input string to a number.
8. (5) The program source code fully complies with the style guide.

6 Advice and Reminders

1. See `errno1.c` in the unit2 sample code for an example of using `strtol`. Note, however, that the example code will **not** report something like "3rd" as an invalid number. Rather, `strtol` will return the number 3 with no error, so you need to account for this.
2. The `isdigit` library function can be used to determine if a character is a digit, returning true or false. See the man page for `isdigit` for how to use it (or search the web). The way to access the `i`'th character within the string `argv[1]` is to use: `argv[1][i]`.
3. To test the success of your program, you can verify the exit/return value by using the command **`echo $?`**, e.g.:

```
$ ./asciitotal hello 3
The ASCII total of the string = 532
The total * multiplier        = 1596
$ echo $?
0
$ ./asciitotal Hello 500
Error: the input multiplier is greater than 100.
$ echo $?
1
```