Computer Science I - Exam 2

ECEN 194

Fall 2022

This midterm consists of 2 programs and is worth 50 points. Each program will be graded with the same rubric as a hack (25 points each, style/documentation/design and correctness).

- You ***may not work with any other students or individuals*** on this exam. You may ask *clarifying* questions on piazza or to Learning Assistants, but they will not provide the same level of help as they would on a lab or hack.

- All work on this exam must be entirely your own. We will be regularly running MOSS, our plagiarism detection software as well as monitoring handin and grader logs for suspicious activity.

- You must hand in your source files using the webhandin and should check your programs using the grader. You may resubmit your programs any number of times up until the due date/time.
- You should perform *basic error checking*
- You are encouraged to write and use any utility or helper functions you wish. To do so, you *must* place all function prototypes (for both exercises) into a single header file, `utils.h` and function definitions into a single source file named `utils.c`. These need to be turned in even if you do not end up using them. Starter files have been provided.

Husker/Maverick Fizz Buzz

1. Write a program that processes a file with the following format: the first line will be a single integer, `n` that indicates how many additional lines the file contains. Each line after the first line will contain a single integer. For example:

```
7
2
9
4
25
57
45
53
```

The program will take the file name to be processed *as a command line argument* and process each of the integer as if it were a "score" and award it to either the huskers team or the mavericks team according to the following rules: - if it is divisible by 3 it should be awarded to the huskers - if it is divisible by 5 it should be awarded to the mavericks - if it is divisible by both 3 *and* 5 it should be awarded to *both* “teams” - otherwise ignore it (it is not given to either team)

For example, with the input file above, - `9, 57, 45` would be awarded to the huskers for a total of 111 - `25, 45` would be awarded to the mavericks for a total of 70 - `2, 4, 53` would all be ignored

Output the two totals to the standard output. Your output may look something like the following.

```
Husker total:    111
Mavericks total:  70
```

Place your full program in a source file named `fizzBuzzFile.c`

2. Someone has scrambled the contents of a text file by 1) reversing every line (except for the endline character) and 2) reversing the order of the lines. For example, a file that looks like this:

```
This is the first line
This is the second line
This is the third line
```

has been scrambled to look like this:

```
enil driht eht si sihT
enil dnoces eht si sihT
enil tsrif eht si sihT
```

Write a program that recovers the original file. Your program should take the input file name as a command line argument, open its contents, reverse each line (except for the endline character), reverse the order of the lines, and output the result to a new file (also taken as a command line argument).

For example, if we were to run your compiled program using:

```
./a.out input.txt output.txt
```

The “scrambled” input file would be `input.txt` and the corrected output file would be `output.txt`. Place your code into a source file named `unscramble.c`.