COMP 3150: Programming in C/C++ (Fall 2021)

Homework 7

Due: Tuesday, October 26, 5:30pm

Reading: Lessons 14, 15, 16

1. [30 points] Write a function with prototype

   ```
   node * deleteTail(node * h);
   ```

   that removes and `free`s the tail node (if any) from the linked list whose head node is pointed to by `h`, and returns a pointer to the head node of the updated linked list. Make your code succinct. Submit your code as `hw7-1-sol.c`.

2. [30 points] Write a program that inputs a university "transcript" from the user in the terminal, and writes the transcript along with the GPA (with three places after the decimal point) to `transcript.txt`. For simplicity, each course is just represented with the number of credits and a letter grade `A`, `B`, `C`, `D`, or `F` (which contribute `4`, `3`, `2`, `1`, or `0` grade points per credit, respectively.) The input is finished when the user enters `0` for number of credits (a sentinel value). For example, if the user enters

   ```
   3 A
   3 C
   4 B
   3 F
   1 A
   3 D
   0
   ```

   then the contents of `transcript.txt` should be

   ```
   3 A
   3 C
   4 B
   3 F
   1 A
   3 D
   GPA: 2.176
   ```

   Don't bother handling when the user inputs invalid data. Hint: To input the letter grade with `scanf`, use `" %c"` as the control string—the space before `%c` makes `scanf` skip over whitespace. Submit your code as `hw7-2-sol.c`.

(more on back)

3. [40 points] Write a "flash cards" app. The card data is in a text file with this format: The first line is the number of cards, `n`. The next `2 * n` lines consist of two lines per card—a *clue* on one line, and the *answer* on the next line. Each clue and each answer is at most fifty characters.

   Use a command line argument to get the name of the file with the card data. Dynamically allocate memory to store the card data. For every card—in a random order—display the clue and read the user's guess for the answer in the terminal. Tell the user whether they entered the correct answer, and if they didn't then display the correct answer. After all `n` cards, tell the user how many they got right out of `n`. For example, if the executable file is `flashcards` and you run `flashcards spanishcolors.txt` then here's a possible execution:

```
yellow
> amarillo
Yes!

blue
> azul
Yes!

red
> rouge
No, it's rojo

green
> verde
Yes!

Your score: 3/4
```

   Also try out `statecapitals.txt`.

   Hints: Assuming `f` points to the `FILE` struct, you can use `fscanf(f, "%zu\n", &n)` to read `n` and go past the subsequent newline in the file. You can use `fgets(clue, 52, f)` to read an individual clue into a `clue` array of `char`s—the `52` includes the `'\n'` and `'\0'`. You can similarly use `fgets(guess, 52, stdin)` to input the user's guess from the terminal. Instead of shuffling the cards themselves (which would involve swapping hundreds of characters), it's more efficient to shuffle an array of indices (for indexing into the cards array). The function `strcmp` from `string.h` returns `0` if and only if the two argument strings are equal. Don't forget `fclose` and `free`.

   Submit your code as `hw7-3-sol.c`.