CS 0449 – Project 1: Blackjack and ID3 Tags

Due: Sunday, June 10, 2018, at 11:59pm

Your first project is to write two programs in C that provide some experience with a wide range of the topics we have been discussing in class.

Blackjack Implementation (30 points)

Blackjack (also known as 21) is a multiplayer card game, with fairly simple rules. For this assignment, you will be implementing a simplified version where a user can play against the computer who acts as dealer.

Two cards are dealt to each player. The dealer shows one card face up, and the other is face down. The player gets to see both of his or her cards and the total of them is added. Face cards (Kings, Queens, and Jacks) are worth 10 points, Aces are worth 1 or 11 points, and all other cards are worth their face value. The goal of the game is to get as close to 21 ("blackjack") without going over (called "busting").

The human player goes first, making his or her decisions based on the single dealer card showing. The player has two choices: Hit or Stand. Hit means to take another card. Stand means that the player wishes no more cards, and ends the turn, allowing for the dealer to play.

The dealer must hit if their card total is less than 17, and must stand if it is 17 or higher.

Whichever player gets closest to 21 without exceeding it, wins.

```
The dealer:
? + 10

You:
4 + 10 = 14

Would you like to "hit" or "stand"? hit

The dealer:
? + 10

You:
14 + 10 = 24 BUSTED!

You busted. Dealer wins.
```

Requirements and Hints

- Have the program intelligently determine if an Ace should be interpreted as a 1 or an 11 by counting the number of aces dealt and adjusting the total down if over 21 and an ace exists.
- Generating random numbers in C is a two-step part. First, we need to seed the random number generator *once* per program. The idiom to do this is:
  ```
  srand((unsigned int)time(NULL));
  ```

- When we need random numbers, we can use the rand() function. It returns an unsigned integer between 0 and RAND_MAX. We can use modulus to reduce it to the range we need:
  int value = rand() % (high - low + 1) + low;
- Remember that getting a card worth 10 is more common because of the face cards, so generate a random card, not a random value.

ID3 Tag Editor (70 points)

An ID3 version 1.1 tag for MP3s (and other multimedia files) adds metadata describing the file. The contents of an ID3 tag are the following:

Offset	length	description
0	3	"TAG" identifier string.
3	30	Song title string.
33	30	Artist string.
63	30	Album string.
93	4	Year string.
97	28	Comment string.
125	1	Zero byte separator.
126	1	Track number byte.
127	1	Genre identifier byte.

An ID3 (v1.1) tag like above is appended to a file as the last 128 bytes. The tag is present if at the -128 from end offset there are the three ASCII characters TAG.

What to Do

For your project you will make a utility that can print the contents of an existing tag, if there, and add or modify a tag.

Make a program called id3tagEd and make it so that it runs with the following command line options:

```
id3tagEd FILENAME
```

should print the contents of the ID3 tag to the console if present, or give a message if not present

```
id3tagEd FILENAME -FIELD VALUE
```

should set the specified field to the value that follows it. You should handle the following field names (specified in lowercase):

- Title
- Artist
- Album
- Year
- Comment
- Track

Make sure you are able to handle multiple {field, value} pairs on the command line at once. If the tag already exists in the specified file, then edit it, if not create a new tag by appending a tag at the end with the specified fields populated. Assume all others are to be empty (filled with 0s.)

Hints and Requirements

- The strings may not have a null-terminator. Be careful that you do the right thing in these cases. Look into the strn family of functions.
- We need to treat these files as *binary files* rather than *text files*. Make sure to open the file correctly, and to use fread and fwrite for I/O.
- Please use a structure to represent an ID3 tag rather than a bunch of disjoint strings. Do your fread() and fwrite() with the whole structure at once. (That is, read or write an entire tag in one file operation.)
- Note that you do NOT need to handle the genre field. Just leave it unchanged if editing, and make it 0 if making a new tag.
- atoi() from <stdlib.h> converts a string to an integer in the fashion of Integer.parseInt()

Environment

Ensure that your program builds and runs on thoth.cs.pitt.edu as that will be where we are testing.

Submission

When you're done, create a gzipped tarball (as we did in the first lab) of your commented source files and compiled executables.

```
~jrmst106/submit/449/
```

*Make sure you name the file with your **username**, and that you have your name in the comments of your source file.*

Note that this directory is insert-only, you may not delete or modify your submissions once in the directory. If you've made a mistake before the deadline, resubmit with a number suffix like `abc123_project1_1.tar.gz`

The highest numbered file before the deadline will be the one that is graded, however for simplicity, please make sure you've done all the work and included all necessary files before you submit