Assignment #1: C-Language Warmup

CS201 Spring 2023
15 points
due Saturday, Feb. 4th, 11:59 pm
second-chance (for 14 points) due Friday, Feb. 10th, 11:59 pm

1 Song Struct

Create a file `warmup.netid.h` (using your netid) and put this in it:

```
#define MAXNAMELEN 63

typedef struct {
  char title[1+MAXNAMLEN];
  char artist[1+MAXNAMELEN];
  unsigned int year;
} Song;
```

2 Functions

Then, create a file `warmup.netid.c` (again, using your netid). Write the two functions described below.

2.1 First function: creating a song

Write a function to allocate and initialize a `Song`:

```
Song *createSong(char *title, char *artist, unsigned int year);
```

If all inputs are valid, then this function will allocate and return a `Song` structure, initialized with the supplied values. All of the following conditions must be true:

- both `title` and `artist` are not `NULL`
- both `title` and `artist` have length > 0 and length $\leq$ `MAXNAMELEN`

If one or more of the conditions is not true, then the function will return `NULL`.

2.2 Second function: comparing two songs

Write a function to compare two songs:

```
int compareSongs(Song *songOne, Song *songTwo);
```

This will return -1, 0, or 1, based on the following:

- if the title of `songOne` precedes (lexicographically) the title of `songTwo`, then return -1
- if the title of `songOne` follows (lexicographically) the title of `songTwo`, then return 1
- if the title of `songOne` is the same as the title of `songTwo`, then
  - if the artist of `songOne` precedes (lexicographically) the artist of `songTwo`, then return -1
  - if the artist of `songOne` follows (lexicographically) the artist of `songTwo`. then return 1
  - if the artist of `songOne` is the same as the artist of `songTwo`, then return -1, 0, or 1, based on comparing the year values of `songOne` and `songTwo`

So in this way, you can use the `compareSongs()` function for sorting a collection of songs: given two songs, it returns an unambiguous answer describing whether the two songs are equal; and if they're not equal, then which song should come before the other in a sorted list.

Use the function `strcmp()` to compare strings. See the example program `strcmp-example.c` in the course gitlab.

3 File Structure

Create two files: `warmup.netid.h` and `warmup.netid.c`, using your UVM netid in place of `netid`.

Put your definitions in `warmup.netid.h` and your code in `warmup.netid.c`. Your .c file should #include your .h file.

Do not ever #include a .c file, and do not ever #include a .h file inside of another .h file.

4 Testing and Development

You'll need to create your own `main()` function as you develop and test your code. In your `main()`, put in calls to your two functions using various inputs to test that your functions are working correctly. But when you're ready to submit, strip out your `main()` function.

Be creative in your testing. Test boundary cases. Make sure that your return values are correct.

Then, get the test code that I've created: `warmup-tests.c`. It's in the gitlab repo for the course: https://gitlab.uvm.edu/Jason.Hibbeler/forstudents_s23, under `CS201/Assignments/Warmup`.

Compile your code either this way:

```
$ gcc -c warmup.netid.c warmup-tests.c
$ gcc warmup.netid.o warmup-tests.o
```

or this way:

```
$ gcc warmup.netid.c warmup-tests.c
```

(Or if you're working in an IDE, put `warmup-tests.c` in your project.)

Run the resulting program. If your functions work correctly, you'll get this output:

```
pass test T1
pass test T2
pass test T3
pass test T4
pass test T5
pass test T6
pass test T7
```

```
pass test T8
pass test T9
pass test T10
pass test T11
pass test T12
pass test T13
all tests pass
```

Otherwise, you'll get an error describing a problem. It's possible that your program will crash due to a pointer problem. In this case, you'll have to figure out what's wrong! The debugger can help.

5 Key Points

Here are the key points and things to remember for this assignment:

- a string in C can be considered to be either a character array or a character pointer
- use `strcpy()` to assign a value to a string
- use `strcmp()` to compare two strings for lexicographic ordering (including for equality)
- know the difference between a string that is a `NULL` pointer and the empty string (“”)

6 What to Submit

Submit two files to Blackboard: `warmup.netid.h` and `warmup.netid.c`.

Use your UVM netid in the filenames. Do not put a `main()` in the .c file that you submit.

7 Graduate Students

Students taking the course for graduate credit, and undergraduates for a bit of extra credit, do the following: get the file `buildSongArray.c` from gitlab. It contains the definition of a function `buildSongArray()`. Call the function in this way:

```
Song *songArray;
int numSongs;
buildSongArray(&songArray, &numSongs);
```

This will allocate and initialize an array of songs. Then, sort the array using by calling the C function `qsort()` (you'll have to read about this function). One of the parameters to `qsort()` will be your `compareSongs()` function. Print the song array before and after you sort it.