Program 3 - Strings and IO

Due Date: 5:00 p.m., October 10 (Monday)

All programs will be tested on the machines in the Q22 lab. It is required that your code run on the computers in the lab (or via ssh).

Part 1 -- String Manipulations

In the third program you will create a library file containing 3 functions, and 2 global variables. I have provided driver code that runs and tests your library code. You will need to create a makefile that compiles both the driver code and your library code to object files, and then links them into an executable.

- Part A
  - For this part of the program you will implement your own version of the library function strstr(), with the following function interface:
    int myStrStr (char haystack[], char needle[], char buffer[]);
    - Your function will take 3 strings, a 'haystack' string, a 'needle' string, and a buffer string. You will search the haystack string for a sequence matching the needle string, and copy the found result (the entire substring) from the haystack string into the buffer (do not copy the needle string).
    - You will return a 1 if the matching sequence in the haystack is found and a 0 if the needle is not found.
    - You can not use the library strstr() function in your implementation. It is only for reference.
  - I will test your function with the following hardcoded strings. **Do not ask for user input.**
    - haystack="apple", needle="app"
    - haystack="orange", needle="ge"
    - haystack="blueberry", needle="ueber"
    - haystack="strawberry", needle="strawberry"
    - haystack="banana", needle="na"
    - haystack="grapefruit", needle="terrible"

Part 2: What's Our Jedi Name

- Part A
  - What is your Jedi Name? To find out, follow these steps:

        - For your Name:
            - Take the first 3 letters of your last name
            - Add the first 2 letters of your first name
        - For example, my jedi name is: Moost
    - You must write a function, findJediName, that takes 3 parameters, char * first, char * last, and a string buffer, then copies your jedi name into the buffer
        - If your last name has less than 3 letters, decide for yourself how to handle the situation
    - Create two global variables (I know I said never to do this, but just this one time it is okay) called, char first_name= “your first name” and char last_name=”your last name”

- ## Part B
    - Using the attached file, names.txt, read in and produce jedi names for each person in our class
    - Read in the last and first name separated by a comma (1 set per line) and store the jedi name in an array of string buffers passed as a parameter
        - It is possible that some people will have more than two names, or names that are not long enough. Since each name is on its own line, you can determine how best to handle this. Write your code with sanity checks and determine for yourself how best to handle names that do not conform to the guidelines.
    - Once you have read all the names, return the return the number of names read from the file.

- ### Extra Credit
    - To generate a true Jedi name, you also need someone’s hometown. For example, my full jedi name would be Moost Cypressian. For extra credit, you should extend the driver code to add the following:
        - Ask the user for her first name, last name, and hometown from stdin.
        - Using the same algorithm as above, generate the Jedi name, except you must add the hometown as a last name with the suffix “ian”.
            - Steven Moore from Cypress becomes Moost Cypressian
        - Write the new jedi name to a file called “jedi.txt”.
    - To get the extra credit you must add a Readme to your submission that states you completed the extra credit.

Part 3 - Submission

- Required code organization:
    - program3.c //driver code
    - strfunctions.c

    - makefile
    - names.txt
    - Readme (if extra credit completed)
- While inside your program 3 folder, create a zip archive with the following command
    - zip -r program3 *
        - This creates an archive of all file and folders in the current directory called program3.zip
        - **Do not zip the folder itself, only the files required for the lab**
- Upload the archive to Blackboard under Program 3.

Grading Guidelines

Total: 20 points

- **Part 1: 12 points**
    - myStrStr function works with all test cases (12 points / 2 for each case)
- **Part 2: 8 points**
    - Generates student's jedi name (2 points)
    - Reads "names.txt" and generates names for all students (6 points)
- **Extra Credit: 5 points** (no partial credit)
    - Works exactly as described
    - Contains a Readme file formatted as described
- **Style Guidelines**
    - Your submission will be rejected if it does not…
        - Pass Valgrind Tests
        - Follow requested program structure and submission format
        - Follow formatting guidelines