CSC 235 PROJECT 4

Points: 30
Objective: Implement functions, properly handle data, and pass parameters.
Requirement: **Update** an x86_64 assembler program that currently inputs data on students, prints the list, and permits search by name. Your project will add a field to each record, use functions, and permit searches.

For your final project you will update the example program *ArrayStructMalloc.m* as follows:

- Add a field for gender, consisting of a single character
  - Add a *define* macro to specify its offset in the record
  - Add instructions to input it after age and before the level
  - Output it when printing the list as *male* or *female*
- Convert it to appropriate modularity by creating the following functions, which perform along the lines of those in *ArrayFnsSearch.m*
  - getList: Read data from the user
    - Parameter: Base address of the list of students, in a register
    - Returns: Number of datum (students) read
  - printList: Print the list of students input.
    - Parameters: Base address of the list of students, number of records, in registers
    - Returns: N/A
  - searchList: Facilitate searching in the array. This function will loop, reading a search key from the user and taking action as specified in the table below
    - Parameters: Base address of the list of students, number of records, in registers
    - Returns: N/A

Length	Operation
0	Terminate search
1	The search key is a gender (m or f, case insensitive. Call *printByGender* to output all students whose gender matches the input.
>1	The search key is a name. Call *search*, which will return the index where that name is found (case sensitive) or a value to indicate that it wasn't found. Print the result.

  - search: Determine the index of the search key
    - Parameters: Base address of the list of students, number of records, address of search key, in registers
    - Returns: Index where found, or value indicating not found
  - printByGender: Print all records whose gender field matched the search key
    - Parameters: Base address of the list of students, number of records, search key (a char), in registers. The search key is in the lowest byte.
    - Returns: N/A

Notes:

- As you are required to use macros, your program must be named **p4.m** . Use asm, not assem
- You will likely want to use some C-string handling functions. Beyond the ones already seen, you may find strchr() to be of interest.
- You may **NOT** ask how many students will be entered. The program will input up to 10 students, terminating on entry of the 10th student or ^D to the name prompt. Programs that ask for how many will receive a grade of 0.

CSC 235 PROJECT 4

- The single bytes in this program (level, gender, search key for *printByGender*()) are accessed and manipulated using appropriate instructions, e.g. **movb**.
    - Use the proper register references and associated instructions, e.g. `%al` for the lowest 8 bits of `%rax`.
- The project will be graded using scripts. Consequently, the order of input of a student's data must be precisely name, age, gender, and level.
    - 2 letter grade penalty for any program that can't be run with the script.
    - Sample scripts that work with *ArrayStructMalloc.m* will be provided.
    - **TEST USING THE SCRIPT. WRITE SCRIPTS OF YOUR OWN!!!!**
- Use of gets and puts stdio functions is permitted.
- All data is stored in frames, except string literals, including those holding input formats. Any other use of global registers or static variables will result in a penalty of **at least** 7 points.
- **You are expected to use macros in your program** as has been done previously. It not only makes it easier for someone else to follow your program, it also makes it easier for you to follow your own program. You will be penalized if you don't use macros. You will also be penalized severely for submitting a program that isn't well formatted, with straight columns delimiting instruction mnemonics and the first operands. Check your formatting on acad using a Unix editor.
    - **Do not use tabs between operands.** Comma and space
    - Limit lines to about 80 characters.
- You must properly save registers in your functions and pass arguments, and anything else, using proper procedures within space you are entitled to.
    - You must follow register use as if your program would be linked with a program compiled using gcc or g++. Arguments must be placed in registers in designated order.
- You are encouraged, in the strongest terms, to develop this program incrementally. That's how the examples were developed. Recommended process:
    - Add gender to record, update code for input and print. **TEST!!!!!**
    - Convert to functions. Test each one as you go.
    - Update the search
    - Add the printByGender function and update searchList to call correct function (when not exiting).

  Back up the working version after each update is completed.
- Document your decisions regarding storage. Beware of leaving things in registers to get clobbered (`%r11` has shown a propensity). Letter grade penalty for submissions that do not document this. Well named macros are part of this equation.

Deliverables: **Deadline:** TBA, in class. Submit late at your own risk policy in effect.

D2L (Project 4 drop box)**:** **Readme**. Specific design decisions and bugs are to be detailed in this file. MUST be readable on D2L.

Turnin: **p4.m**