Assignment #8

Introduction to C Programming – COP 3223

Objectives

1. To learn how to design and implement functions for program development
2. To reinforce how to use pass by value and pass by reference variables
3. (Optional) To learn how to use structures

Problem: Dragon Collecting (dragondex.c)

Movies about dragons and dragon training were very popular this summer. Your friend has not stopped talking about how awesome dragons are and how cool it would be to train them. After completing your dragon training simulator, your friend is ready to graduate to having a team of dragons.

You are now building a program that will store a collection of dragons for your friend's team. Your friend will need to be able to do the following:

1) Add a dragon to his team
2) Remove a dragon from his team
3) Search for a particular dragon
4) List all of the dragons of a particular color on the team

To make the testing of the program easier, your program will read in all of its input from a file called "dragon.txt". Your friend's team of dragon cannot exceed 100 dragons.

Header Specification

To facilitate grading, include in your header comment which portions of the assignment you have implemented. Your choices are as follows:

(1) Adding dragons only
(2) Adding and removing dragons only
(3) Adding, removing, and searching dragons only
(4) All options implemented: adding, removing, searching, and listing dragons

If your comment is accurate, meaning that you pass the appropriate tests cases corresponding to your choice, you'll earn 10 points. If your comment is roughly accurate, meaning that you sincerely attempted the items you listed, and most of them work minus a tiny bug, then you'll get 5 of these points. If your comment isn't accurate to a reasonable degree, you'll get 0 of these points.

The reason this is here is because it's very important to communicate to others accurately what you've accomplished and what is left to accomplish. This sort of honesty and ability to appraise your own work is a critical skill in a job.

Input Specification (dragon.txt)

The first line of the file will contain a single positive integer representing the number of updates to the dragon team. (**Note:** We assume that at the beginning of the program, the team has no dragons on it.)

Each following line will have directions for a single update to the dragon team. The formats for these lines are as follows:

The very first value on each of these lines will be a string from the following subset: {ADD, REMOVE, SEARCH, LIST}. This string indicates which operation (corresponding to the list in the problem statement) is being designated.

If the operation is to add a dragon (ADD), then the line will contain the following information after ADD, separated by spaces:

NAME COLOR

The name and color of the dragon will be strings that contain only alphabetic letters and underscores. None of these strings will exceed 39 characters. No dragon will share a name with another dragon, though there may be multiple dragons of the same color.

If the operation is to remove a dragon (REMOVE), then the line will only contain the exact name of the dragon after REMOVE.

If the operation is to search for a dragon (SEARCH), then the line will only contain the exact name of the dragon that is being searched for.

If the operation is to list all of the dragons by a particular color (LIST), then the line will only contain the desired color after LIST.

Output Specification

All output should go to the screen. Separate the output for each command with one blank line. Here is the format for output for each command:

For adding a dragon, simply write out a statement of the form:

X the Y dragon has been added to the team.

For removing a dragon, simply write out a statement of the form:

X the Y dragon has been removed from the team.

For both of these, X represents the name of the dragon and Y represents the dragon's color. You may assume that remove operations in the input file will always be valid. Thus, if a remove is ever requested, the dragon in question is guaranteed to be on the team.

For searching for a dragon, output one of two statements, depending on whether or not the designated dragon was found in the library:

X the dragon is currently on the team.
X the dragon is NOT currently on the team.

For searching for all the dragons of a particular color, output a single line of the form:

Y dragons:

Y represents the color requested. Each following line should list the name of one dragon of that color on the team. List each dragon exactly once.

Implementation Restrictions

You must use the following constants:

```
#define MAX_LENGTH 40
#define MAX_TEAM 1000
```

Though function prototypes will not be provided, it is expected that you follow good programming design and create several functions with well-specified tasks related to the solution of this problem. Make sure to pay very careful attention to parameter passing.

Bonus Points

10 bonus points are possible with this program, making a perfect score 110/100. In order to gain these points, you must use the following structures to store information:

```
struct dragon {
    char name[MAX_LENGTH];
    char color[MAX_LENGTH];
};

struct collection {
    struct dragon team[MAX_TEAM];
    int num_dragons;
};
```

It is not sufficient to simply have the structures in your program, you must use them store information and process operations for the dragon team.

Hints

If you choose to use the above structures, use pass by reference parameters for your functions. This will ensure that any change made to the dragon team in the function is reflected in main. Inside the function, remember to access the components using the -> syntax for structures.

Use the prewritten string function when dealing with strings:

- use **strcpy** whenever you want to copy the contents of one string into another, instead of the equals sign.
- use **strcmp** to determine which operation is being requested.

Whenever you add a dragon to the team, make sure you add it to the first open slot on the team.

Whenever you remove a dragon from the team, make sure you copy the dragon in the last spot on the team into the vacated spot. For example, if the dragon to be removed is in index 3 and the number of dragons on the team before removal is 7, then the dragon in index 6 (the last filled index) should be copied into the dragon spot in index 3. Subsequently, the number of dragons on the team should be updated to 6.

<u>Sample Input File (library.txt)</u>

```
12
ADD TOOTHFUL BLACK
ADD SPITFIRE RED
SEARCH FURY
ADD FURY RED
ADD ICICLE BLUE
ADD NIGHTWING BLACK
LIST BLACK
REMOVE SPITFIRE
ADD STARFALL BLUE
LIST RED
LIST GREEN
LIST BLUE
```

<u>Sample Output (corresponding to input file)</u>

```
TOOTHFUL the BLACK dragon has been added to the team.

SPITFIRE the RED dragon has been added to the team.

FURY the dragon is NOT currently on the team.

FURY the RED dragon has been added to the team.

ICICLE the BLUE dragon has been added to the team.

NIGHTWING the BLACK dragon has been added to the team.

BLACK dragons:
TOOTHFUL
NIGHTWING

SPITFIRE the RED dragon has been removed from the team.

STARFALL the BLUE dragon has been added to the team.

RED dragons:
FURY

GREEN dragons:

BLUE dragons:
ICICLE
STARFALL
```

<u>Deliverables</u>

One source file: *dragondex.c* for your solution to the given problem submitted over WebCourses.

Restrictions
Although you may use other compilers, your program must compile and run using Code::Blocks. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details
Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.