**Programming Assignment 3 -- CS429 Fall 2013**

Due Date: 13 October 2013

**Purpose**

The purpose of this assignment is mainly to write a C program using pointers. You will need to write the program (facts.c), compile it, and then execute it. You can execute it with several different tests of your own, and when you are satisfied, turn the source in to the TA for grading (use "turnin").

**Motivation**

In this class, we deal with a lot of facts. Facts about lots of different types of computers. It hurts to try to remember all of them. So let's write a program to remember them all for us.

So what is a fact? It's a piece of information about a particular property of an object. For example,

* Fact: The word-size of the CDC6600 is 60 bits.
* Fact: The PDP-8 has 8 instructions.
* Fact: The PDP-8 has a 12-bit address.
* Fact: The x86 architecture has 8 registers.
* Fact: The x86\_64 architecture has 16 registers.
* Fact: The x86\_64 architecture has a 64-bit address.
* Fact: Lauren's cat's name is Sasha.

We want a program to remember all these facts, and then help us answer questions, like

* Question: What is the address for a PDP-8?
* Question: How many registers does the x86 have?
* Question: What is Lauren's cat's name?

**Program Specification**

Your problem is to write a program to read a set of facts and remember them. It should then read a set of questions, look up the answers, and print them. So, you should write a program "facts.c" which takes two files for input: (1) a set of facts and (2) a set of questions.

The facts file will have a sequence of lines. Each line will start with an "F" (to indicate that this is a fact), and then three strings: (1) the name of an object, (2) a property of that object, and (3) the value of that property for that object. The syntax is

F [object-name]: [property]=[value]

The colon (:), equal sign (=) and newline are the delimiters. Ignore any lines that are blank, or start with "#" instead of "F".

The questions file will have a sequence of lines. Each line will start with a "Q" (to indicate that this is a question), and then two strings: (1) the name of an object, and (2) a property of that object, separated by a colon. Again, ignore any lines that are blank, or start with "#" instead of "Q". The colon (:) and newline are the delimiters.

The output for a question will be one line that looks like a fact, and is printed to standard output.

If the second file (the questions file) is not given, read your questions from standard input.

**Example**

Assume your input facts were (computers.fax):

F CDC6600: number\_registers=24 F CDC6600: opcode\_bits=6 F PDP11: number\_registers=8 F PDP11: opcode\_bits=4,8,10 F IBM360: number\_registers=16 F IBM360: opcode\_bits=8

and your input questions were (computers.q):

Q PDP11: number\_registers Q CDC6600: number\_registers Q IBM360: opcode\_bits

Then running "facts computers.fax computers.q" will produce output that looks like:

F PDP11: number\_registers=8 F CDC6600: number\_registers=24 F IBM360: opcode\_bits=8

**Error Cases**

Consider the following error cases:

* Failure to provide the facts file -- print an error message and quit.
* Cannot open file %s -- print the error message with the file name and exit.
* First character is not an "F" or "Q".
* Bad syntax for a fact -- missing the colon or the equal sign of a Fact.
* Same fact with different values -- keep the most recent value for a property.
* Question asks about an object-name or property that we do not have a fact for -- print "F object-name property=unknown".
* Providing more (or less) than one or two file names; you always need the facts file, but the questions can either be a file or stdin.

You will need to use malloc and free, and pointers, for this assignment. If you run out of memory, print "No more memory", and exit.

**Test Cases**

Provided test cases are

* [computers.fax](http://cs.utexas.edu/users/peterson/prog3/computers.fax)and [computers.q](http://cs.utexas.edu/users/peterson/prog3/computers.q)which produces [computers.a](http://cs.utexas.edu/users/peterson/prog3/computers.a).

**Submission**

You need to submit a single C file named facts.c to program3. More specifically, for submission you should use

turnin --submit zxy program3 facts.c

Your program should compile with "gcc -Wall facts.c" with no errors or warnings.

We may also run your program under "valgrind"; valgrind should find no memory leaks.

**Extensions**

Extensions to consider:

* Can you merge the Facts and Questions file into one file, allowing new facts to be added on the fly?
* Can you allow other types of questions, like "What objects have this property?" or "What objects have this property and value"?
* This is effectively a small data-base. How would you extend it to a full data-base?
* This is the beginning of an artificial intelligence program. What would you need to do to make it more natural and intelligent?

Due Date: 13 October 2013