1. Write a function fact\_calc that takes a string output argument and an integer input argument n and returns a string showing the calculation of n!. For example, if the value supplied for n were 6, the string returned would be “6!=6x5x4x3x2x1=720”. Write a program that repeatedly prompts the user for an integer between 0 and 9, calls fact\_calc and outputs the resulting string in a box of asterisks of the right size to surround the result. If the user inputs an invalid value, the program should display an error message and reprompt for valid input. Input of the sentinel -1 should cause the input loop to exit. Here is a sample run:

Enter an integer between 0 and 9 or -1 to quit => 6

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 6!=6x5x4x3x2x1=720 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter an integer between 0 and 9 or -1 to quit => 12

Invalid entry

Enter an integer between 0 and 9 or -1 to quit => 0

\*\*\*\*\*\*\*

\* 0!=1 \*

\*\*\*\*\*\*\*

Enter an integer between 0 and 9 or -1 to quit => -1

1. Declare a structure with two members real and imaginary of type double for complex numbers. Write a function named add\_complex to add two complex numbers (both complex structures) and return the result (another complex structure). Write a test program that prompts the user to enter two complex numbers and show the sum of the two input complex numbers. Here is a sample run:

Enter two complex number: 4.3+2.1i 6.2+1.5i

The sum is 10.5+3.6i

1. Write a program that asks the user to enter a fraction, then reduces the fraction to the lowest term. Here is an example run:

Enter a fraction: 6/12

In lowest terms: 1/2

Declare a fraction structure, which contains two members: numerator and denominator (both of type int). In the program, write the following function:

* void reduce(struct fraction f, struct fraction \* reduced\_f)

f is a fraction and reduced\_f is a pointer to a structure variable in which the function will store the fraction once it has been reduced to lowest terms.

[Hint: To reduce a fraction to lowest terms, first compute the GCD of the numerator and denominator. Then divide both the numerator and denominator by GCD. ]

1. A number expressed in scientific notation is represented by its mantissa ( a fraction) and its exponent (an integer). Define a type sci\_not\_t that has separate components for these two parts. Define a function scan\_sci that takes from the input source a string representing a positive number in scientific notation, and breaks it into components for storage in a sci\_not\_t structure. The mantissa of an input value (m) should satisfy this condition: 0.1 <= m < 1.0. Also, write functions to compute the sum, difference, product, and quotient of two sci\_not\_t values. All these functions should have a result type of sci\_not\_t and should ensure that the result’s mantissa is in the prescribed range. Define a print\_sci function as well. Then, create a driver program to test your functions. Your output should be of this form:

Value input: 0.25000e3 0.20000e1

Sum: 0.25200e3

Difference: 0.24800e3

Product: 0.50000e3

Quotient:0.12500e3

1. Numeric addresses for computers on Internet are composed of four parts, separated by periods, of the form

xx.yy.zz.mm

where xx, yy, zz, and mm are positive integers. Locally, computers are usually known by a nickname as well. Please design a program to process a list of Internet addresses, identifying all pairs of computers from the same locality. Create a structure type called address\_t with components for the four integers of an Internet address and a fifth component in which to store an associated nickname of ten characters. Your program should read a list of up to 100 addresses and nicknames terminated by a sentinel address of all zeros and a sentinel nickname.

**Sample Data**

111.22.3.44 platte

555.66.7.88 wabash

111.22.5.66 green

0.0.0.0 none

The program should display a list of messages identifying each pair of computers from the same locality, that is each pair of computers with matching values in the first two components of the address. In the messages, the computers should be identified by their nicknames.

**Example message**

Machines platte and green are on the same local network.

Follow the messages by a display of the full list of addresses and nicknames. Include in your program a scan\_address function, a print\_address function, and a local\_address function. Function local\_address should take two address structures as input parameters and return 1 (for true) if the addresses are on the same local network, and 0 (for false) otherwise.