STAT 740 - Spring 2004 - Homework 1

Due: Friday, January 30th

Note: Problems 1 and 2 require computer code. A hardcopy is not needed. Please e-mail me the (uncompiled) code so that I can compile and run it to verify that it works. Neither of these two programs involve issues of efficiency, nor do they require additional comment lines except for the date of last modification and your name.

1) The `accov.for` program we've seen in class does not currently force the user to enter an integer when asked for the item numbers. Because of this the program could be caused to crash by entering a non-integer or by entering an integer with a too-large or too-small value. A subroutine called `chartoint` is available on the course web-page (currently demonstrated inside of another program). This subroutine is designed to ask the user to enter a non-negative integer value in a specified range, and then to force them to do it.

For this problem, include this subroutine in accov.for and modify the program to use the subroutine. Note that you should also change the 999 to quit option to a more logical non-negative integer choice as well.

2) We saw in class what happens to the `INTEGER*1` format when values are accumulated one at a time (e.g. 127+1=-128). How does the computer deal with cases of multiplication though? Write a program to calculate 64x2, 64x3, 64x4, 64x5, 65x2, 65x3, 65x4, and 65x5 and store the values in the `integer*1` format and then print them out. Can you tell how it seems to be handling the multiplication? (Binary work is not required.)

3) Imagine we were working with an `INTEGER*3` variable using two's complement notation. What would the largest (most positive) and smallest (most negative) values representable by this notation be? Give the numbers and their representations.

4) Represent 3 and 1/16 in IEEE single precision floating point representation like we used in class.

5) Determine the value of the largest number representable in IEEE single precision floating point by calculating the decimal value corresponding to:

S=0 E=11111110 F=11111111111111111111111

You can report in scientific notation with at least four decimal places instead of carrying it all the way out.

6) Imagine that you want to add a list of several thousand real numbers of various magnitudes together. In what order would you add them together to guarantee that you would get the smallest rounding error possible? Give a brief example using a short list of numbers. (Note: No programming needed, and the answer isn't just to add them from smallest to biggest!)