Name: ______________________________ Section: __________

ID#: ______________________________

CSE/MATH 455, Arnold | Lab 1 | September 6, 1996

Some more exploration of finite precision arithmetic

Problem 1 (only) is to be handed in on Wednesday, September 11. In addition to this page, hand in a listing of the two small programs (or email the programs to `dna@math.psu.edu`). Your written explanations should be clear, well thought out, and well written.

1. (a) Write a program to read a number n and compute the sum $\sum_{j=1}^{n} \frac{1}{j^2}$. Use single precision (so don't use Matlab, which is all double precision). Using your program complete the following table.

n	10	100	1,000	10,000	100,000
sum					

What value does this suggest for the infinite series $\sum_{j=1}^{\infty} \frac{1}{j^2}$? __________

(b) Write a second version of the program which differs in that the terms are summed from smallest to largest: $j = n, n-1, \ldots, 1$. Give the results of this program:

n	10	100	1,000	10,000	100,000
sum					

Apparent value of $\sum_{j=1}^{\infty} \frac{1}{j^2}$: __________

(c) The infinite series is in fact known to be exactly equal to $\pi^2/6$. Which program for computing the sum appears to be more accurate? On a separate sheet discuss why this is the case.

(d) It is helpful to think of an extreme case. What would be the result of computing the following sum in single precision floating point arithmetic in the order given, and in reverse order?

$$100,000,000. + \underbrace{1. + 1. + \cdots + 1.}_{1,000 \text{ times}}$$

(You shouldn't need the computer for this, although you can use it to check.) Explain.

2. Matlab has a function `chop(x,n)` which rounds (not chops!) the number x to n decimal places. Familiarize yourself with this function by computing, e.g., `chop(pi,2)`, `chop(pi,3)`, $\ldots$. Give the command `format long` to see 15 decimal places.

Use the following Matlab statement to generate a list of 1,000 random floating point numbers between 0 and 20:

```
x = 20*rand(1,1000);
```

Chop all the numbers in the list to four decimal places:

```
xapprox = chop(x,4);
```

Compute the relative error: `relerr=(x-xapprox)./x;`. Theoretically, what is the largest magnitude the relative error can be? Check this in two ways: `max(abs(relerr))` and `plot(relerr)`.

Now examine how the relative error introduced by `chop` gets propagated by some functions. Take the square root of all the numbers in the list `x`, and see how well the square root of `xapprox` approximates them:

```
s=sqrt(x); sapprox=sqrt(xapprox); srelerr=(s-sapprox)./s;
```

Plot the relative error in the square root. Put the new plot on top of the previous plot by typing `hold on` before plotting. You'll need to plot in a different color to see the new plot. Use, e.g., `plot(srelerr,'r')` will plot in red. (Do `help plot` for more colors.) How do the relative errors in the x and $\sqrt{x}$ compare?

Study some other functions similarly: x^2, x^3, $\exp(x)$, $\ldots$. Try to explain what you see.

3. The following simple Fortran 90 program is to compute $\int_a^b f(x)\,dx$ using the right endpoint rule with n interval. I.e., to compute

$$h[f(a+h)+f(a+2h)+f(a+3h)+\cdots+f(b)]$$

where $h=(b-a)/n$. What's wrong with the program? What will happen when it is run?

```
c define a, b, and n
      a = 1.
      b = 4.
      n = 30
c compute h; initialize a and sum
      h = (b-a)/n
      x = a
      sum = 0.0
c loop through sum, stopping when we hit b
      do
        x = x+h
        sum = sum + f(x)
        if (x == b) then
          print*,h*sum
          stop
        endif
      end do
c
      end
      function f(x)
      f = sin(x)*exp(x)
      return
      end
```