Eric Olson

§2.1#4 Predict the output of

```
I = 0
X = 0
H = 0.1
While X < 1.0
    I = I + 1
    X = X + H
    disp([I, X])
end
```

when executed on a binary computer that uses chopping. How many times did the programmer intend for this loop to be executed. Does the intended behavior actually occur. What happens? Would the outcome be different if $X = X + H$ were replaced by $X = I * H$?

Since $0.1 = \frac{1}{10} = \frac{1}{2 \times 5}$ is a fraction whose denominator has prime factorization consisting of primes other than powers of 2, then 0.1 has a repeating dyadic expansion. In particular, in binary ten is 1010_2 and

$$
\begin{array}{r}
.0001100 \\
1010_2 \overline{)\,1.0000} \\
\quad 1010 \\
\quad\quad 1100 \\
\quad\quad 1010 \\
\quad\quad\quad 1000
\end{array}
$$

Chopping means that H will contain a number slightly smaller than 0.1.

§2.1 #4 continues...

Since H contains a number smaller that 0.1 it will take 11 iterations of the while loop before X accumulated to a value greater than or equal 1.0.

Logically the program should terminate after only 10 iterations, assuming that exact arithmetic is being used. This is probably what the programmer had in mind.

If X=X+H is replaced by X=I*H, the same behavior occurs because chopping is used. If round to the nearest were used, then X=I*H would result in 10 iterations.

§2.145 the following MATLAB program produced the given output. Explain the results

```
x = 0.0
while x < 1.0
   x = x + 0.1;
   disp([x, sqrt(x)])
end
```

x	0.1	0.2	0.3	0.4	0.5	0.6
$\sqrt{x}$	0.3162	0.4472	0.5477	0.6325	0.7071	0.7746

x	0.7	0.8	0.9	1.0	1.1
$\sqrt{x}$	0.8367	0.8944	0.9487	1.0000	1.0488

This loop went for 11 iterations, the reason is similar to why the loop in §1.4 ran for 7 iterations. However, note that MATLAB is using the standard rounding mode & round to the nearest representable number. Thus when $\frac{1}{10}$ is represented with a rounded double precision 52-bit with a hidden bit we have

```
1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
1  0  0  1  1  0  0  1  1  0  0  1  1  0  0  1  1  0  0  1  1  0  0  1  1  0  0

28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
1  1  0  0  1  1  0  0  1  1  0  0  1  1  0  0  1  1  0  0  1  1  0  1  0
                                                                  ^
```

and the number is, by chance, rounded up. Thus it is slightly more than $\frac{1}{10}$. Rounding gives us the 5th iteration that $x = 0.5$ exactly. However

§2.1#5 continues...

In binary, the sum .5 + .1 is given by

```
  .1
  1000 11001 1001100...
  .1001100 11001100...
```

and when this is rounded to 52 bits with one hidden bit it rounds down. Subsequent additions have the same effect. Thus, at the 10th iteration the value in X is slightly less than 1.0.

§2.21. Calculate the error, relative error, and number of significant digits in the following approximations.

(a) $x_T = 28.254$, $x_A = 28.271$

$\text{Error}(x_A) = x_T - x_A = -0.017$

$\text{Rel}(x_A) = \frac{x_T - x_A}{x_T} = \frac{-0.017}{28.254} \approx -0.000601684712$

$|\text{Rel}(x_A)| \leq 5 \times 10^{-m-1}$

$.00060168 = 5 \times 10^{-m-1}$

$\log_{10} \frac{.00060168}{5} = -m-1$

So $m = \left(\log_{10} \frac{5}{.00060168}\right) - 1 \approx 2.9196$ and we have that x_A has about 2.9 significant digits. By inspection we see x_A is good to 3 significant digits

(d) $x_T = \sqrt{2}$, $x_A = 1.414$

$\text{Error}(x_A) = x_T - x_A = \sqrt{2} - 1.414$

So $0.0002135 \leq \text{Error}(x_A) \leq 0.0002136$

$\text{Rel}(x_A) = \frac{x_T - x_A}{x_T} = \frac{\sqrt{2} - 1.414}{\sqrt{2}}$

§2.2 #1(d) continues...

$$0.00015101 \leq Rel(x_4) \leq 0.00015102$$

Significant digits is about

$$m = \left(\log_{10} \frac{5}{0.00015102}\right) - 1 \approx 3.52.$$

By inspection there are 4 significant digits.

§2.2 #5 acd. Use identities to rewrite the following expressions to avoid loss-of-precision.

(a) $$\frac{1-\cos x}{x^2} = \frac{(1-\cos x)(1+\cos x)}{x^2(1+\cos x)}$$

$$= \frac{1-\cos^2 x}{x^2(1+\cos x)} = \frac{\sin^2 x}{x^2(1+\cos x)}.$$

(c) $$\sin(a+x) - \sin a = \sin a \cos x + \cos a \sin x - \sin a$$

$$= (\sin a)(\cos x - 1) + \cos a \sin x$$

$$= -\sin a \frac{(1-\cos x)(1+\cos x)}{1+\cos x} + \cos a \sin x$$

$$= -\sin a \frac{(1^2 - \cos^2 x)}{1+\cos x} + \cos a \sin x$$

$$= -\frac{\sin a \sin^2 x}{1+\cos x} + \cos a \sin x$$

$$= \frac{\sin x}{1+\cos x}\left(\cos a + \cos a \cos x - \sin a \sin x\right)$$

$$= \frac{\sin x}{1+\cos x}\left(\cos a + \cos(a+x)\right)$$

§2.2 #5 continues...

(d) $\sqrt[3]{1+x} - 1 = \left(\sqrt[3]{1+x} - 1\right) \dfrac{(1+x)^{2/3} + (1+x)^{1/3} + 1}{(1+x)^{2/3} + (1+x)^{1/3} + 1}$

$= \dfrac{1+x-1}{(1+x)^{2/3} + (1+x)^{1/3} + 1} = \dfrac{x}{(1+x)^{2/3} + (1+x)^{1/3} + 1}$

MATH/CS 466/666 FALL 2008 HOMEWORK 2

§2.2#12. Computationally, examine the accuracy of the identity

$$\sin(x) = \sqrt{1-\cos^2(x)}, \qquad 0 \le x \le \frac{\pi}{2}$$

for values of x near 0. Take x approaching 0 and examine the relative error of $\sqrt{1-\cos^2(x)}$ by comparing it to $\sin(x)$ computed directly.

We take values of $x = 1/2^n$ with $n = 0, 1, \ldots, 12$ using the C language program

```
#include <stdio.h>
#include <math.h>

int main(){
    int i;
    double x=1.0;
    printf("Math/CS 466/666 Fall 2008\n");
    printf("\nSection 2.2 Problem 12:\n");
    printf(" %22s %22s %22s\n","x","Error","Rel");
    for(i=0;i<=12;i++){
        double lhs=sin(x);
        double t=cos(x);
        double rhs=sqrt(1.0-t*t);
        printf(" %22.15e %22.15e %22.15e\n",
            x,lhs-rhs,(lhs-rhs)/lhs);
        x/=2.0;
    }
    return 0;
}
```

with output

```
Math/CS 466/666 Fall 2008

Section 2.2 Problem 12:
                     x                   Error                     Rel
  1.000000000000000e+00  0.000000000000000e+00  0.000000000000000e+00
  5.000000000000000e-01  1.110223024625157e-16  2.315736095030427e-16
  2.500000000000000e-01 -1.942890293094024e-16 -7.853109137575391e-16
  1.250000000000000e-01 -3.747002708109903e-16 -3.005422675765572e-15
  6.250000000000000e-02 -5.273559366969494e-16 -8.443190782642927e-15
  3.125000000000000e-02  1.096345236817342e-15  3.508875836023206e-14
  1.562500000000000e-02 -2.059116765984470e-15 -1.317888354589891e-13
  7.812500000000000e-03 -6.947567521287112e-15 -8.892976891010442e-13
  3.906250000000000e-03 -8.213048297012193e-15 -2.102545711081292e-12
  1.953125000000000e-03 -1.891303953727164e-14 -9.683482399674208e-12
  9.765625000000000e-04 -3.790359952987599e-14 -3.881329208779935e-11
  4.882812500000000e-04 -7.579061076651294e-14 -1.552191770176744e-10
  2.441406250000000e-04 -1.515824412604699e-13 -6.208816855707902e-10
```

As $x \to 0$ the approximation becomes less and less accurate because of loss of precision that results from subtracting two numbers nearly equal to 1.

3d, #4 Find bounds for the error and relative error in approximating $\sin(\sqrt{2})$ by $\sin(1.4142)$.

With a calculator:

$$\text{Error}(\sin(1.4142)) = \sin(\sqrt{2}) - \sin(1.4142)$$

implies $0.0000333 \leq \text{Error}(\sin(1.4142)) \leq 0.0000334$

$$\text{Rel}(\sin(1.4142)) = \frac{\sin(\sqrt{2}) - \sin(1.4142)}{\sin(\sqrt{2})}$$

implies $0.0000337 \leq \text{Rel}(\sin(1.4142)) \leq 0.0000338$

Without a calculator:

$$|\text{Error}(\sin(1.4142))| = |\sin(\sqrt{2}) - \sin(1.4142)|$$

$$= |\cos(c)||\sqrt{2} - 1.4142|$$

$$\leq |\sqrt{2} - 1.4142| \leq .0005.$$

Now, since $\pi/6 \leq \sqrt{2} \leq 5\pi/6$ then $\sin\sqrt{2} \geq 0.5$. Hence

$$|\text{Rel}(\sin(1.4142))| = \left|\frac{\text{Error}(\sin(1.4142))}{\sin(\sqrt{2})}\right|$$

$$\leq \frac{.0005}{.5} = 0.001.$$

Note the bounds done without the calculator are not as precise.

MATH/CS 466/666 FALL 2008 HOMEWORK 2

§2.4#1abc. Write a computer program to evaluate the sums

$$\sum_{j=1}^{n} \frac{1}{j(j+1)} = \frac{n}{n+1}$$

$$\sum_{j=1}^{n} \frac{1}{j(j+2)} = \frac{3}{4} - \frac{2n+3}{2(n+1)(n+2)}$$

$$\sum_{j=1}^{n} \frac{1}{\sqrt{j(j+1)}[\sqrt{j}+\sqrt{j+1}]} = \frac{n}{\sqrt{n+1}[\sqrt{n+1}+1]}$$

Do the sum by SL smallest to largest and by LS largest to smallest. Calculate a true value for the sum using the formula and compare it with the values obtains by LS and SL.

Consider the C language computer program

```
#include <stdio.h>
#include <math.h>

int N[]={ 10,50,100,500,1000,5000 };

double a(int j){ return 1.0/j/(j+1); }
double A(int n){ return (double)n/(n+1); }

double b(int j){ return 1.0/j/(j+2); }
double B(int n){ return 3.0/4-(double)(2*n+3)/2/(n+1)/(n+2); }

double c(int j){
    register double t=(double)j;
    return 1/sqrt(t*(t+1))/(sqrt(t)+sqrt(t+1));
}
double C(int n){
    register double t=sqrt((double)(n+1));
    return (double)n/t/(t+1);
}

struct series {
    double (*x)(int), (*X)(int);
    char c;
} s[] = {
    a,A,'a',b,B,'b',c,C,'c' };

int main(){
    int i;
    printf("Math/CS 466/666 Fall 2008\n");
    for(i=0;i<sizeof(s)/sizeof(struct series);i++){
```

```
31          int k;
32          printf("\nSection 2.4 Problem 1%c:\n",s[i].c);
33          printf(" %5s %22s %22s %22s\n",
34              "n","sum","error-LS","error-SL");
35          for(k=0;k<sizeof(N)/sizeof(int);k++){
36              int j,n=N[k];
37              double ans=s[i].X(n),sumLS=0.0,sumSL=0.0;
38              for(j=1;j<=n;j++){
39                  sumLS+=s[i].x(j);
40              }
41              for(j=n;j>=1;j--){
42                  sumSL+=s[i].x(j);
43              }
44              printf(" %5d %22.15e %22.15e %22.15e\n",
45                  n,ans,ans-sumLS,ans-sumSL);
46          }
47      }
48  }
```

with output

```
Math/CS 466/666 Fall 2008

Section 2.4 Problem 1a:
    n                    sum                  error-LS                  error-SL
   10  9.090909090909091e-01  0.000000000000000e+00 -1.110223024625157e-16
   50  9.803921568627451e-01  4.440892098500626e-16 -1.110223024625157e-16
  100  9.900990099009901e-01  3.330669073875470e-16  0.000000000000000e+00
  500  9.980039920159680e-01  0.000000000000000e+00  0.000000000000000e+00
 1000  9.990009990009990e-01 -5.551115123125783e-16  1.110223024625157e-16
 5000  9.998000399920016e-01  2.220446049250313e-16  0.000000000000000e+00

Section 2.4 Problem 1b:
    n                    sum                  error-LS                  error-SL
   10  6.628787878787878e-01  0.000000000000000e+00 -1.110223024625157e-16
   50  7.305806938159879e-01 -3.330669073875470e-16  0.000000000000000e+00
  100  7.401475441661813e-01 -1.110223024625157e-16  0.000000000000000e+00
  500  7.480059800717290e-01 -9.992007221626409e-16  0.000000000000000e+00
 1000  7.490014975044915e-01 -6.661338147750939e-16  0.000000000000000e+00
 5000  7.498000599800072e-01 -2.442490654175344e-15  0.000000000000000e+00

Section 2.4 Problem 1c:
    n                    sum                  error-LS                  error-SL
   10  6.984886554222364e-01  0.000000000000000e+00  0.000000000000000e+00
   50  8.599719915971989e-01 -2.220446049250313e-16  0.000000000000000e+00
  100  9.004962809790011e-01 -1.110223024625157e-16  1.110223024625157e-16
  500  9.553232948391229e-01 -4.440892098500626e-16  1.110223024625157e-16
 1000  9.683930229379493e-01  0.000000000000000e+00  1.110223024625157e-16
 5000  9.858592783777349e-01 -1.443289932012704e-15  2.220446049250313e-16
```

In almost all cases the SL sum added from smallest to largest is the most accurate.