CS 100M: Lecture 6
February 10

P2 Challenge Problems on the website by mid-afternoon.

for k=1:N
x = rand;
if k == 1
% Initialize maxVal
maxVal = x;
else
if x > maxVal
maxVal = x;
end
end

Mistakes!

for k=1:N
x = rand;
if k == 1
% Initialize maxVal
maxVal == x;
else
if x > maxVal
maxVal = x;
end
end

Correction!

Typical While Set-up

< Initializations >
while < Boolean condition >

Loop Body

end

< stuff to do after the loop >

Sum the first 10 Integers

k = 0; s = 0;
while k < 10

k = k+1;
s = s+k;
end

fprintf('1+2+...+10 = %2d',s)

Eg.1: Mantissa Length

Visualize a calculator window that reports results in scientific notation with 6-digit mantissas:

.1 0 0 0 0 0 x 10^1

If we add very small number to this, there may be no change
Discovering mantissa length

Idea. Look at these sums until no change:

\[
\begin{align*}
1 + 1 & \quad \text{change} \\
1 + 1/10 & \quad \text{change} \\
1 + 1/100 & \quad \text{change} \\
1 + 1/1000 & \quad \text{change} \\
1 + 1/10000 & \quad \text{change} \\
1 + 1/100000 & \quad \text{no change (needs a 7-digit mantissa)}
\end{align*}
\]

Counting the number of steps where there is a change will give you the Mantissa length.

Switch to base-2

\[
\begin{align*}
j &= 0; \\
y &= 1; \\
\text{while } 1+y > 1 & \quad j = j+1; \\
\text{ } & \quad y = y/2; \quad \% y = 1/2^j \\
\text{end}
\end{align*}
\]

At the close of the loop, the value of \( j \) is the smallest integer so computed value of \( 1 + 1 / 2^j \) is one.

Eg.2: Diverging Series

The sum

\[
s_n = 1 + 1/2 + 1/3 + \ldots + 1/n
\]

can be made arbitrarily large by choosing \( n \) big enough.

Given \( A \) (positive), what must \( n \) be so \( s_n > A \) ?

Solution Framework

Initialize running sum and an index to keep track of the terms. Get \( A \).

\[
\text{while (running sum is still too small)}
\]

Update running sum and index

\[
\text{end}
\]

Print final index value and final sum

\[
s = 0; \\
n = 0; \\
A = \text{input(‘Sum Threshold:‘)}
\]

while \( s < A \)

\[
\begin{align*}
n &= n+1; \\
s &= s + 1/n;
\end{align*}
\]

end

fprintf( etc )

Eg.3 A Newton Fractal

Matlab supports complex arithmetic

\[
i = \sqrt{-1}; \\
z = 3 + 4i; \\
z = (3z^4 + 1)/(4z^3); \\
\text{if abs(z - i) < 10^{-6};} \\
\text{disp(‘Close to i’);}
\]

end
An Interesting Sequence

For any complex $z_0$ that is not on the 45 degree lines in the complex plane, the sequence $z_0, z_1, z_2, \ldots$

$$z_{n+1} = \frac{(3z_n^4 + 1)}{4z_n^3}$$

converges to either 1, i, -1, or -i.

Examine Convergence

Enter a starting $z$ and see how long it takes to "get close" to one the four "attractors" 1, i, -1, -i.

A while loop setting: don't know how many steps it'll take.

```matlab
tol = .001;
i = sqrt(-1);
[x,y] = ginput(1);
z = x + i*y;
its = 0;
while (abs(z-1)>tol) && (abs(z-i)>tol) && (abs(z+1)>tol) && (abs(z+i)>tol)
    z = (3*z^4 + 1)/(4*z^3);
    its = its+1;
end
fprintf('Number of iterations = %1d',its)
```

Eg.4 : “First to 21”

Two players: H and T.
H scores a point if a coin toss is heads.
T scores a point if a coin toss is tails.
First to score 21 points wins.

```matlab
nHeads = 0;
nTails = 0;
while (nHeads < 21) && (nTails < 21)
    x = rand;
    if x<=.5
        fprintf('H');
        nHeads = nHeads + 1;
    else
        fprintf('T');
        nTails = nTails + 1;
    end
end
```
**Analysis of “First to 21”**

On average, how many total tosses are there in a game?

**Nested Loops**

The body of a loop can contain another Loop.

**Example**

```
sum = 0; N = 1000;
for k = 1:N
    for k = 1:N
        Another Loop can be part of this
    end
end
ave = sum/N;
```

Play a game of “First to 21”

Add to sum the total number of tosses required to play the game.