Simulations using Mathematica

One part of this class will involve simulations using the software package Mathematica. (You are free to use other programming/software packages, but solutions to simulation problems will be in Mathematica.) First you must start Mathematica. I use Linux so I type

```
math
```

to get the line version of Mathematica or type

```
mathematica
```

to get the notebook version of Mathematica. The notebook version is easier to work with (editing is easier), but sometimes with remote logins this is slow so I resort to the line version. Since I'm doing this remotely, the examples I'll give will have the format of the line version, but the commands are the same.

After typing one of these commands, Mathematica responds with

```
Mathematica 4.2 for Linux
Copyright 1988-2002 Wolfram Research, Inc.
 -- Motif graphics initialized --

In[1]:=
```

The symbol `In[1]:=` is input line #1 and is waiting for you to type a command. I then typed $3 + 5$ and here's what happened:

```
In[1]:= 3+5

Out[1]= 8
```

If I had ended the `In[1]:=` line with a semicolon, then the output line would have been surpressed. You might ask why you would want to do this. Well, the answer could be very long and you don't want to see it printed out but, of course, you want to record the answer for subsequent use so you give it a name. Here are some examples

```
In[2]:= su=3+5;

In[3]:= Sqrt[su]

Out[3]= 2 Sqrt[2]

In[4]:= su+su
```

```
Out[4]= 16

In[5]:= su!

Out[5]= 40320
```

Mathematica does all sorts of things and you should get the manual or at least consult the website listed in the Math 131 Homepage for some on-line help. For example, Mathematica is very good at typical calculus problems. In this class we will mainly use Mathematica as a programming language to carry out simulations of stochastic processes. For programming I find it most convenient to write programs in another file and then read them into a Mathematica session. This way when I make a mistake (which is common!), I can easily edit the file and then re-read the file. This saves lots of typing.

Suppose we wish to simulate the toss of a fair coin. To do this we will use Mathematica's (pseudo) random number generator `Random[]`. At the command level here is what happens

```
In[5]:= Random[]

Out[5]= 0.906977
```

Mathematica returns a number x, $0 \leq x \leq 1$ that is "uniformly distributed" on the interval $[0, 1]$. We can use this to simulate coin tossing. If x lies in the interval $[0, 1/2)$ we call the result 1 (heads) and if it lies between $[1/2, 1]$ we call the result 0 (tails). Here is the function that does this

```
In[8]:= coin[]:=If[Random[]<1/2,1,0];
```

To find out about the command `If` you type

```
In[9]:= ?If
If[condition, t, f] gives t if condition evaluates to True, and f if it
   evaluates to False. If[condition, t, f, u] gives u if condition evaluates
   to neither True nor False.
```

I can now make a list of outcomes of tossing the coin ten times. I use the `Table` command to produce the list

```
In[10]:= outcomes=Table[coin[],{i,1,10}]

Out[10]= {1, 0, 1, 0, 0, 1, 1, 1, 1, 1}
```

I could now, for example average this set of numbers

```
In[11]:= Sum[outcomes[[i]],{i,1,10}]/10

         7
Out[11]= --
         10
```

Note; if `list` is a list then `list[[i]]` is the ith member of the list. We might want to do this more than ten times. I repeat the simulation $10,000$ times.

```
In[13]:= outcomes=Table[coin[],{i,1,10000}];

In[14]:= Sum[outcomes[[i]],{i,1,10000}]/10000

         2517
Out[14]= ----
         5000

In[15]:= N[Out[14]]

Out[15]= 0.5034
```

Note: The semicolon at the end of `In[13]` is very useful! To convert `Out[14]` to decimal form I used the `N` command (numerical).

Let's now write a program `birthdayProblem` to simulate the birthday problem. This program uses the `Module` construction

```
In[3]:= ?Module
Module[{x, y, ... }, expr] specifies that occurrences of the symbols x, y, ...
    in expr should be treated as local. Module[{x = x0, ... }, expr] defines
   initial values for x, ... .
```

Here is the program I wrote

```
(* This problem simulates the birthday problem  *)
(* n= number of people in sample *)

birthday[n_]:=Module[{i,blist={},distinctBList},
                For[i=1,i<=n,i++,blist=Append[blist,Random[Integer,{1,365}]]];
                distinctBList=Union[blist];
                If[Length[blist]>Length[distinctBList],1,0]];

(* This program uses the Module construction of a function.
We initialize the birthday list to the empty list.  Then we use
the For loop construction to append to this list a random
```

```
integer between 1 and 365.  (We are neglecting leap years.)
The command Union returns the list of distinct elements in
the bList.  If the length of blist is greater than the length
of distinctBList, then in the sample of n people there is
a coincidence of birthdays.  We return the value 1 if this happens;
otherwise, we return the value 0
*)

bSimulation[noPeople_,noSimulations_]:=
        Module[{i,total=0},
        For[i=1,i<=noSimulations,i++,
                total=total+birthday[noPeople]];
        N[total/noSimulations]];

(* This function carries out the simulations.  We initialize
total to zero and then we call birthday[noPeople] for
noSimulations times.  One is added to total if there is a
coincidence of birthdays otherwise zero is added.  After
this is done we divide total by the noSimulations and take
the numerical value.  This value is then returned.
*)
```

To use this program I first start MATHEMATICA and the first command is to read in file `birthdayProblem`. I then call `bSimulation` for 65 people (class size) and for 5,000 simulations of the birthday problem. The fraction of these simulations that had a birthday coincidence is 0.9984.

```
Mathematica 4.2 for Linux
Copyright 1988-2002 Wolfram Research, Inc.
 -- Motif graphics initialized --

In[1]:= <<birthdayProblem;

In[2]:= bSimulation[65,5000]

Out[2]= 0.9984

In[3]:=
```