Maple Labs for Discrete Mathematics

Kate McGivney and Doug Ensley
kgmcgi@ship.edu and deensl@ship.edu
Shippensburg University

Lab 1. Introduction to Maple for Discrete Math

Part 1. Maple syntax: basics, plotting and help files

Maple is a computer algebra system used in industry, science, engineering, and, of course, many of your math courses at Ship. Like any good computer application, there are a few little idiosyncrasies to get used to (like "*Maple* requires each command to end with ; or :" and "*Maple* is case-sensitive"). Once you get used to it, I hope you will find *Maple* to be an invaluable tool in all of your mathematical pursuits.

Put your cursor anywhere within the *Maple* input, and hit the Enter key.

```
> 1 + 2 + 3^2;
```

Notice that we ended the line with a semicolon. Remove the semicolon and hit Enter again to see the message you get. When you see that message, you should know that you forgot to end the command with a semicolon.

```
>
```

One fun thing you can do in *Maple* is compute LARGE numbers. Ever wonder how many digits $2^{1000000}$ has? Me too! You might regret it, but if you feel adventurous, put your cursor on the next line and hit Enter.

```
> 2^1000000;
```

Ok, it was a mistake to execute that last command. Since we only wanted to know the length of this large number, we could have instead given a name to 2^1000000, and then asked *Maple* how long the resulting number was. The trick is to not let *Maple* show the answer to 2^1000000 as an interim step. We can do this (i.e., ask *Maple* to keep an answer to itself) by ending the line with a colon instead of a semicolon. For example, execute the next two lines:

```
> n := 2^1000000:
> length(n);
```

We used a colon after the first line because we did not want to see the result of that command, but we used a semicolon after the command "length(n)" because we did want to see the answer to that.

Notice that the command "n := 2^1000000" (that's a colon followed by an equal sign with no space in between) assigned the value 2^1000000

to the variable *n*. Assignments like this come in handy any time you want to refer back to an expression later or when you want to keep expressions as simple as possible. For example, the following assigns the expression 2^x to the name *f*, and the number $\sqrt{2}$ to the name *a*.

```
> f := 2^x;
> a := sqrt(2);
```

The *Maple* command *evalf* is probably the most commonly used command, so you might as well start now. It merely returns a floating point evaluation of the input.
We can now use the *evalf* command to see a floating point evaluation of *a*.

```
> evalf(a);
```

Another pretty natural thing to want to do is to substitute the number *a* into the expression *f*. Most people think that this should be written *f*(*a*), but if you try that below you will see that you get a weird answer, not what you wanted.

```
> f(a);
```

There is a *Maple* command which does the correct thing though. To remember it, you must remember that what you really want to do here is to take the expression whose name is *f* and substitute for the letter *x* in it the value of *a* . Here are two examples of the **subs**titution command.

```
> subs(x = a, f);
> subs(x = sqrt(2), x^2);
```

TRY THIS. Did you notice that *Maple* did not give you a number for the answer in the first case above? Go back and wrap the **evalf( )** command around each the first expression above to see what the numerical value is. Remember that the semicolon should still come at the end of the line.

Plotting

One of the most useful features of *Maple* is its graphics capabilities. The syntax for plotting is pretty simple. For example, try the following:

```
> plot(x^2 - 4*x - 50,x=0..10);
```

So the plot command requires only two arguments: The first is the expression to be plotted, the second gives the independent variable (*x*)

and the range of its values (0 to 10) over which the expression's graph will be plotted. Before continuing, answer the following questions about this graph.

TRY THIS. What is the exact value (to two correct decimal places) of *x* shown where the graph crosses the *x*-axis? What will be the other value of *x* where the graph crosses the *x* axis? (Think about it, and then change the command accordingly to see if you are right.)

In Discrete Math, we often work with sequences, where for each entry there is a next entry (and not necessarily any in between). In *Maple*, this is called a "point plot," and it has a syntax very similar to the command above.

```
> plot([[1,2], [3, 4], [2, 5]], x=0..4, y=0..5,
style = point, color=blue);
```

The square brackets above are what *Maple* uses to denote a "list". We will visit this structure many times in this course.

Help files

The single most important thing to be able to do in *Maple* is **USE THE HELP FILES**. This is because *Maple* has more capabilities than can ever be addressed in a single course, so you are best off learning how to figure out how to make *Maple* do any new thing you want through judicious use of its built-in help files. You can get into the help files through the Help Browser in the menus, but it is often hard to find what you want there. A better thing to do is to type `? cmdname`. This brings up the specific help file for cmdname which includes syntax, examples and related commands. For example, suppose we want to compute the average value of the first 100 positive perfect square numbers (1, 4, 9, 16, etc.). To do this we need to add up the first 100 perfect squares and divide by 100. Start with asking for help with "add" with the command line below, and then try to figure out how to get the answer to this question.

```
> ?add
```

TRY THIS. The average of the first 100 positive perfect squares is 3383.5, which is not a whole number. Find three values of *n* so that the average value of the first *n* positive perfect squares is a whole number. What do your values have in common?

Part 2. Functions

One thing that computers force you to do is to be consistent with your choice of defined structures. We saw above that we got gibberish when we defined an expression *f* and tried to compute *f*(*a*). Let's see it again here:

```
> f := 2^x;
> f(3);
```

We solved the problem before by pointing out that we are really trying to "substitute 3 for *x* in *f*" and so we used the appropriate **subs** command to do what we meant to do. The math notation *f*(3) is reserved for use when *f* is a function, not an expression. There is a difference as we shall see. To define *g* to be the function that takes a number *x* as input and returns the number 2^x as output, we make the following assignment:

```
> g := x -> 2^x;
```

Note that we can now use functional notation to refer to putting a number or expression as input into a function definition.

```
> g(3);
> g(p-1);
```

This type of definition is appropriate only when we have an algebraic expression (i.e., a closed formula) for the numbers we are trying to generate. We have seen in Section 1.2 however, that it is often easier to find a recursive formula instead. We finish this introduction by showing how to implement such a description in *Maple*.

Part 3. Procedures

Maple is actually a fully-supported programming language which happens to have extensive built-in mathematics functions and symbolic algebra capabilities. Instead of being a compiled language, *Maple* is interpreted, which means that errors are checked as the commands are being executed. Consequently, Maple's error messages are not as helpful as you might be used to.

Here is an example of a procedure that simply performs the same task the function *g* above.

```
> g := proc(x)
   RETURN(2^x)
end;
> g(3);
```



```
> g(p-1);
```

The procedure below is intended to compute entries in the sequence 3, 7, 11, 15, ... in which each term is 4 more than the previous term. Write a recursive description of this sequence first, and then try to see how each component of your definition are realized in the procedure below. (**Note**: To get the input spread over several lines, I simply held the shift key down while hitting Enter at the end of the first three lines, and then hit Enter alone after the fourth line to execute the entire block.)

```
> a := proc(n)
option remember:
if (n < 2) then RETURN(3) else RETURN(a(n-1) +
4) fi
end;
> a(1);
> a(2);
> a(3);
> a(4);
> a(5);
```

We conclude this introduction by presenting one more command that allows us to see many values of a sequence, whether defined by procedure or function, so that we can investigate its patterns. The command **seq** creates a list of numbers according to a rule over a specified range of input numbers. For example, the following command lists the first five values of the sequence *a* that we computed above one at a time.

```
> seq( a(i), i=1..5);
```

And the next command lists the first ten positive perfect squares:

```
> seq( i^2, i=1..10);
```

TRY THIS. Alter the above definition of *a* so that you get each of the following sequences. Use the **seq** command to check.

- 1, 3, 7, 15, 31, ...
- 2, 6, 10, 14, 18, ...
- 2, 5, 9, 14, 20, ...

Lab 2. Sequences

In this lab activity we'll use Maple to discover a closed formula given a recursive sequence and vice-versa.

Consider the sequence $a_n = a_{n-1} + 2n + 1$ where $a_1 = 1$ for $n \geq 2$. Using the following Maple script, we can quickly see the first 20 terms of this sequence.

```
a:=array(1...20):
a[1]:=1:
print(1,a[1]);
for i from 2 to 20 do
    a[i]:=a[i-1]+(2*i+1):
    print(i,a[i]);
od:
```

What do you think is the closed form for this sequence?

Check your conjecture by modifying the above script so that the closed form replaces the recursive form.

For each of the following recursive sequences modify the above script so that it produces the first 20 terms of each sequence. Next, make a conjecture about the recursive sequence's closed form. Check your conjecture by modifying the script.

$a_n = 4a_n - 1; a_1 = 2.$

Closed Form:_____

$a_n = 2a_n + b; a_1 = 1.$

Closed Form: _____

For each of the following sequences given in closed form, modify the above script to produce the first 10 terms. Then make a conjecture as

to the sequence's recursive form. Finally verify that your conjecture is valid by modifying the script so that it contains the recursive form.

$a_n = 3^n + 1.$

Recursive Form: _____

$a_n = 3n^2 + n.$

Recursive form: _____

$a_n = 2^{2n-1} - 1$

Recursive form: _____

Lab 3. Recursion and Induction

We have seen a couple of examples already of recursive thinking. The tool we will use to prove things about recursively defined structures is called "the Principle of Mathematical Induction." This lab will illustrate the connection while allowing you to practice the proof technique. You will need *Maple* for the computer part and pencil and paper for the writing part. It is also helpful to have a partner with whom you can talk things through.

Enter the following new function definitions and then answer (among yourselves) the question that follows. (*NOTE:* To get a new line without beginning *Maple* execution, hit "Shift Enter" at the end of the line.)

```
> F := proc(n)
options remember;
  if (nc < 3) then RETURN(1)
    else RETURN(F(n-1)+F(n-2))
  fi
end;
```

```
> S := proc(n)
options remember:
  if (n = 0) then RETURN(0)
     else RETURN(S(n-1) + n)
  fi
end;
```

We will discuss the "options remember" part of these procedures when we discuss recursive algorithms later in the course. To use these new commands, you simple treat them like any other *Maple* commands. For example, execute the next few lines, and then once you've seen the general syntax you can explore some more on your own. The problem you should be investigating is simple:

Problem 1. Describe in words what *F* and *S* compute.

```
> F(3);
> F(4);
> F(5);
```



```
> S(2);
> S(3);
> S(4);
```

Problem 2. Copy the definition of *S* and edit it to create a new recursive function called *oddSum* that given an input of *n* returns the sum of the first *n* odd numbers.

Problem 3. What problem that we worked on already is addressed by the following recursive function? (The function *isPowerofTwo* is not a built-in *Maple* function. It is defined by the first line below, and it returns **true** when its argument is an integer power of 2 and *false* otherwise.)

```
> isPowerofTwo := x ->
evalb(type(simplify(log[2](n)),integer));
```

```
> B := proc(n)
options remember;
  if isPowerofTwo(n) then RETURN(1)
    else RETURN(B(n-1) + 2)
  fi
end;
```

Lab 4. Introduction to Sets

In this activity we will explore various set operations. Given the universal set U:={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} use the following Maple commands to generate random subsets A, B, and C of size 3, 5, and 7, respectively.

```
> with(combinat, randcomb);
> randomize();
> U:={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
> A:=randcomb(U,3);
> B:=randcomb(U,5);
> C:=randcomb(U,7);
```

Use the randomly generated sets to determine the following.

$A \cup B =$

$(A \cup C) \cap B =$

$A \cap B =$

$A' =$

$A \cap (B \cap C) =$

$A - B =$

$A' \cap B =$

$P(A) =$

$AxB =$

Next we'll see how we can use Maple to check our answers. To find the union of sets A and B, simply type


```
> A union B;
```

The intersection command is `A intersect B;`
To find the set A-B, type

```
> A minus B;
```

Maple does not have a built in "complement" command. How can you use one of the above commands to find the complement?

The power set of a set A can be found by typing the following commands:

```
> with(combinat, powerset);
> powerset(A);
```

Lastly, to find the Cartesian product of two sets, A and B, use the following Maple commands.

```
> `union`( op(map(y -> map(x->[x,y],A),B)) );
```

Randomly generate sets A, B, and C again. Answer the following questions *without Maple* and then use Maple to check your answers.

$A \cup B =$

$(A \cup C) \cap B =$

$A \cap B =$

$A - B =$

$A' =$

$A \cap (B \cap C) =$

$A' \cap B =$

$P(A) =$

$AxB =$

Are each of the following properties true or false? We'll use Maple to generate some conjectures. For those that are not true, provide a specific counterexample.

1. $A \cap (B \cup C) = (A \cap B) \cup C$

2. $A - (B \cup C) = (A - B) \cup (A - C)$

3. $P(A) \cup P(B) = P(A \cup B).$

Lab 5. Using Maple for Counting

Maple has a combinatorics package that has many nice tools to help us learn about counting. Execute the following command to load the combinatorics package:

```
> with(combinat);
```

Don't forget that you can ask for help on any of these commands to find out more about them. For example, we will use the *permute* command, look at the help file by executing the following line. Once you have looked it over, select "Close Help Topic" under the File menu, and you will be returned to where you left.

```
> ?permute
```

So for example, if want to list all permutations of length 2 with entries taken from {a, b, c, d}, we would use the following command:

```
> permute([a,b,c,d],2);
```

The command *nops* will count the number of items in any list for you. So the command nops(permute([a, ,b, c, d], 2)); should return the number of permutations in the above list. Try it below.

Problem 1. Draw a "game tree" with two branchings that generates the above list. Can you see how the number of branches is related to the number of permutations generated? Test your hypothesis by thinking about the structure of the game tree for *permute([a,b,c,d,e,f,g,h,i,j],2)*, predicting the number of permutations that will be generated, and testing your prediction using the *nops* command.

Unfortunately Maple does not have a similar command that generates all ordered lists, so we have to write one. Execute the block of code below to define the new command, *orderedLists*, which we will use next.

```
> orderedLists := proc(S,k)
local i, L:
L := NULL:
for i from 1 to nops(S) do
   L := L, seq(S[i],j=1..k)
od:
RETURN(permute([L],k))
end;
> orderedLists([a,b,c,d],2);
```

Problem 2. Draw a "game tree" with two branchings that generates the above list. Can you see how the number of branches is related to the number of ordered lists generated? Test your hypothesis by thinking about the structure of the game tree for *orderedLists([a,b,c,d,e,f,g,h,i,j],2)*, predicting the number of ordered lists that will be generated, and testing your prediction using the *nops* command.

Lab 6. Combinatorial Equivalence

Claim: The following two questions have the same answer.

- How many 2-element subsets of the set {1,2,3,4,5} are there?
- How many 3-element subsets of the set {1,2,3,4,5} are there?

Why is this? Use Maple to generate all possible 2-element and 3-element subsets of {1,2,3,4,5}. Here is the appropriate command:

```
> choose([1,2,3,4,5],n);
```

where *n* is the size of the subset.

Based on the output, can you explain the answer to a beginning discrete math student?

How many 2-element subsets of the set {1,2,3,4,5,6} are there?
How many 4-element subsets of the set {1,2,3,4,5,6} are there?

In general, explain why the number of k-element subsets of the set $\{x_1, x_2, \ldots, x_n\}$ is always equivalent to the number of n-k element subsets of the set $\{x_1, x_2, \ldots, x_n\}$.

Claim: The following two questions have the same answer.

- How many ways are there to flip 3 heads in 5 tosses of a coin?
- How many 3-element subsets of {1,2,3,4,5} are there?

We'll use Maple to investigate why this is true.

In the first case, we want to determine the number of ways to flip 3 heads in 5 tosses. In other words, we want to determine the number of ways to order 3 heads and 2 tails. Therefore the appropriate Maple command is

```
> permute([H,H,H,T,T],5);
```

Generalize the previous result: The number of ways to flip k heads in n tosses is the same as _____.

Lab 7. Permutations and Combinations

In this lab we'll develop a formula that relates the number of subsets and the number of permutations that can be formed from n distinct objects. Recall that the difference between subsets and permutations is that in the first case we don't care about the order of the entries and in the second case we do.

Helpful counting discrete math Maple commands are bundled together. To access these commands, type:

```
> with(combinat);
```

The Maple command `permute(n,r)` generates the permutations of size r from a set of n distinct elements. Recall that our text used the notation P(n,r) to represent the *number* of permutations of size r from the set of n objects.

Execute `permute(n,r);` for the following choices of n and r.

n	3	3	3	3
r	0	1	2	3

What is P(n,r) in each case?

Next, we'll compute the number of subsets of size r from a set of n distinct elements. Since now we do not care about the order of the entries, [1,2] and [2,1] are considered "equivalent". Count the number of subsets that arise in each of the above cases.

n	3	3	3	3
r	0	1	2	3
Number of Subsets				

The Maple command > `choose(n,r)` constructs the subsets of size r that can be formed from n objects. Our text represents the *number* of subsets of size r that can be formed from n objects by the notation C(n,r).

n	3	3	3
r	0	2	3
P(n, r)			
C(n, r)			

Now let's look at a slightly greater set, one that contains 4 distinct entries. Use the `permute(n,r)` and the `choose(n,r)` commands to find all permutations (subsets) of size r. Fill in the number of permutations (subsets) for each choice of r in the following table:

n	4	4	4	4	4
r	0	1	2	3	4
P(n, r)					
C(n, r)					

Can you use the previous tables to come up with a formula that relates P(n, r) and C(n, r) ? Explain why this formula makes sense.

Lab 8. More on sets and permutations

In this lab we will explore the difference between the structures *permutations* and *sets*. We will look at formulas for counting each type of object, and then we will try some more sophisticated investigations using the computers.

First execute the following command in order to load some helpful Discrete Math-ish *Maple* commands.

```
> with(combinat);
```

Now compare the output of the following commands. How much longer is the first output list than the second? Think of how you would explain this precise difference to a classmate.

```
> permute(5,3);
> choose(5,3);
```

Technically, one of the above commands lists permutations and the other lists sets. Since Maple uses the square brackets [and] for the output in both, it is not obvious which is which. Can you tell based on whether order matters or not?

In the text, we use the notation $P(5,3)$ for the number of permutations of length 3 with entries taken from $\{1,2,3,4,5\}$, and the notation $C(5,3)$ for the number of sets of size 3 with entries taken from $\{1,2,3,4,5\}$. (In other words, $C(5,3)$ counts 3-element subsets of $\{1,2,3,4,5\}$.) Using the *nops* command (remember how it tells you how many things are in a long list), compare $P(7, 3)$ and $C(7, 3)$. Is this what you expected?

We can now write (as we do in the text) formulas for P and C. We have already talked about the theoretical basis for these in the last class, but now we want to be sure we understand what P and C are counting and how they are related. The following functions are not built into *Maple* so you will have to define them before you ever use them.

```
> P := proc(n, k)
  RETURN(product(n-j, j=0..(k-1)))
end;
> C := proc(n, k)
   RETURN(P(n,k)/k!)
end;
> P(5, 3);
> C(5, 3);
```


Try this! Use the C and P functions we just defined to explore which of the following statements are true.

(1) $\mathbf{C}(n, k) = \mathbf{C}(n - 1, k) + \mathbf{C}(n - 1, k - 1)$
(2) $\mathbf{P}(n, k) = \mathbf{P}(n - 1, k) + \mathbf{P}(n - 1, k - 1)$
(3) $\mathbf{P}(n, k) = n * \mathbf{P}(n - 1, k - 1)$
(4) $\mathbf{C}(n, 0) + \mathbf{C}(n, 1) + \mathbf{C}(n, 2) + \ldots + \mathbf{C}(n, n) = 2^n$

The Binomial Theorem

One of the important ideas throughout the course is the fact that there is great benefit in understanding many different answers to problems because from that understanding comes the ability to tell how different questions are related to one another.

The distributive law of multiplication over addition tells us that when we expand (a0 + a1)*(b0 + b1) we form terms by choosing each possibility from the first factor and each possibility from the second factor and multiplying them together. The terms so formed are then added together. Look at the expansion below to see if this makes sense:

```
> expand( (a0 + a1)*(b0 + b1));
```

Now expand the following to see how the distributive law generalizes when you multiply three things together.

```
> expand( (a0 + a1)*(b0 + b1)*(c0+c1));
```

How many terms in the above expression have two "1"s in them? How many terms with two "1"s will be in the expansion of $(a0 + a1)*(b0 + b1)*(c0 + c1)*(d0 + d1)$? Check below.

The following question is related to the previous one by using powers of 0 and 1 on a single variable x. (Look for the analogy to the previous discussion.)

If we expand $(x^0 + x)(x^0 + x)(x^0 + x)$, what will be the coefficient of x^2?

In the expansion of $(x^0 + x)(x^0 + x)(x^0 + x)(x^0 + x)$?

```
> expand( (x^0 + x^1)^3);
```

If we formally make an analogy to binary sequences in the comparable expansions we will have explained ...

The Binomial Theorem. The coefficient of x^k in the expansion of $(1+x)^n$ is **C**(*n*, *k*).

Pascal's Triangle

We can make a nice table of values by placing C(n,k) in Row **n**, Entry **k** of our table. Since we have to allow for values like C(5,0), we will think of rows starting with "Entry 0" instead of "Entry 1," and in the same spirit, we will call the top row, Row 0 rather than Row 1. In this fashion, the values in the first three rows of our table will come from the following:

```
> C(0,0);
> C(1,0);
> C(1,1);
> C(2,0);
> C(2,1);
> C(2,2);
```

These can be filled in our table in a triangular shape as shown below. This is the first part of what is known as Pascal's Triangle.

```
      1
    1   1
  1   2   1
```

(1) Fill in the next two rows of Pascal's Triangle.

(2) Express the visual pattern between rows of Pascal's Triangle in words.

(3) Express the same pattern using the **C** notation.

(4) Make a conjecture about the sum of the values in Row *n* of Pascal's Triangle.

(5) State your conjecture using the **C** notation.

Lab 9. Recursive counting

In this lab we will look at some recursive counting techniques and implement them with recursive Maple programs. You will not have to write code from scratch, but you will need to be able to read and modify existing code.

Example 1. Why is P(n, k) = n * P(n - 1, k - 1)?

Consider the question, "How many permutations of length k use entries from {1, 2, ..., n}?" We know the answer is P(n,k), but let's see another way to get this value.

To answer this, we can use a TWO-step decision process:

(1) Choose a first entry from {1, 2, ..., *n*}

(2) Fill in the remaining *k* - 1 entries with a permutation of length *k* - 1 using entries from the *other n* - 1 elements of the set

We can analyze this algorithm using the product rule: There are clearly *n* choices for the first step, and by definition of P, there are P(n-1,k-1) ways to do the second step. Hence by the product rule, there are n * P(n - 1, k - 1) permutations generated by the algorithm. Therefore this is the same thing as P(n,k).

Notice that this relationship does not help if k = 0, so it is also necessary to specify that P(n, 0)= 1 for this recursive formula to make sense.

The following Maple program implements this idea:

```
> P := proc(n, k)
option remember:
if (k = 0) then
  RETURN(1)
fi;
RETURN( n * P(n - 1, k - 1));
end;
```

(A) Compare P(6, 2) or P(10, 3) with the result we would get using our regular formula from the text.

(B) We saw before that numbers like P(2, 5) make sense to ask about -- they simply have a value of 0. (That is, there are 0 permutations of length 5 with entries taken from {1, 2}.) Does the above procedure deal with these unusual cases correctly? If not, how do you fix it?

Example 2. Why is C(n, k) = C(n - 1, k) + C(n - 1, k - 1)?

Consider the question, "How many binary sequences of length n use exactly k 1's?" We know the answer is C(n,k), but let's see another way to get this value.

To answer this, we can use a TWO-step decision process:
(1) Choose a first entry from {0,1}
(2) Fill in the remaining n - 1 entries with a binary sequence of length n - 1.

The trouble is that the number of ways to do step 2 DEPENDS on whether we took a 0 or a 1 in step 1, so to remedy this, we use cases:

CASE I. Suppose the first entry is a 0. In this case, there is one way to do step 1, and then in step 2, we must be sure that our n - 1 digit binary sequence has exactly k "1"s -- there are C(n - 1, k) ways to do step 2. By the product rule, there are C(n - 1, k) binary sequences like this. (i.e. with n digits, k of which are 1s, and starting with a 0).

CASE II. Suppose the first entry is a 1. In this case, there is one way to do step 1, and then in step 2, we must be sure that our n - 1 digit binary sequence has exactly k - 1 "1"s -- there are C(n - 1, k - 1) ways to do step 2. By the product rule, there are C(n - 1, k - 1) binary sequences like this. (i.e. with n digits, k of which are 1s, and starting with a 1).

Hence by the sum rule, there are C(n - 1, k) + C(n - 1, k - 1) binary sequences generated by this algorithm. Therefore this is the same thing as C(n,k).

Notice that this relationship does not help if k = 0 or if k > n, so it is also necessary to specify that C(n, 0) = 1 and that C(n, k) = 0 if k>n for this recursive formula to make sense.

The following Maple program implements this idea:

```
> C := proc(n, k)
option remember:
if (k = 0) then
  RETURN(1)
fi;
if (k > n) then
  RETURN(0)
fi;
RETURN( C(n - 1, k) + C(n - 1, k - 1));
end;
```

Compare the value of C(6, 3) and C(10, 2) with this procedure to what you get from the formulas in our text.

Example 3. How many n digit binary sequences do not have adjacent 1's?

Let b(n) represent this number. Obviously b(1) = 2 because both possible binary sequences of length 1 fail to have adjacent 1's. On the other hand, b(2) = 2 since only the sequences 00, 10 and 01 qualify. What is b(3)? What is b(4)? Do you see a pattern?

Claim: The number of binary sequences of length n having no adjacent 1's is b(n-1) + b(n-2).
To see this, imagine the two step algorithm for forming binary sequences of length n having no adjacent 1's: (1) Choose a first digit, (2) choose the remaining digits as a binary sequence of length n - 1 without adjacent 1's. The problem is that step (2) might not work when a 1 is chosen in step (1), so once again, we break the problem into cases.

Case 1. If the first digit of the sequence is a 0, then there's 1 way to do step (1) and b(n-1) ways to do step 2. Hence there are b(n-1) binary sequences of length n having no adjacent 1's and beginning with a 0.

Case 2. If the first digit of the sequence is a 1, then the only allowable sequences to use in step 2 are those that start with a 0. By the argument in Case 1, there are b(n-2) binary sequences of length n - 1 having no adjacent 1's and beginning with a 0, so this is the number of ways to complete step 2 in this case. This means there are b(n-2) binary sequences of length n having no adjacent 1's and beginning with a 1.

Adding the numbers for the two cases together gives us the number of binary sequences of length n without adjacent 1's. That is, b(n) = b(n-1) + b(n-2).

We can implement this in Maple as follows:

```
> b := proc(n)
if n=1 then RETURN(2) fi;
if n=2 then RETURN(3) fi;
RETURN(b(n-1)+b(n-2));
end;
```

Exercises.

(1) Modify the above argument to count the number of binary sequences of length n which do not have THREE consecutive 1's.

(2) How many ordered lists of length 5 with entries from {1,2,3,4,5,6} have entries summing to 20? (Hint: Let d(n,k) = the number of ways to have a list of length n with entries summing to k, and show that d(n, k) = d(n-1, k-1) + d(n-1, k-2) + d(n-1, k-3) + d(n-1, k-4) + d(n-1, k-5) + d(n-1, k-6). Then implement this in Maple and compute d(5, 20).)

Lab 10. Probability Simulations: The Birthday Problem

What is the probability that at least two of the 32 students in the class share the same birthday?

The following script produces a random set of *k* values from a set of *n* objects. We'll use it to simulate the birthday for each of the 32 students. What are *n* and *k* in this situation?

```
> randlist := proc(n,k)
  local i;
  randomize();
  RETURN(map(rand(1..n),[seq(i,i=1..k)]))
  end proc;
```

Simulate the birthday problem one time. Was there at least one match?

To get an estimate of the likelihood that there will be at least one match we need to perform many simulations. Each group will now simulate this experiment 20 times, keeping track of whether there was at least one match each time. After everyone is done, we'll pool our data to determine the empirical probability that there's at least one match.

To more easily determine whether or not there is a match, we'll sort the data each time. To perform the 20 simulations, type:

```
>sort(randlist(n,k));
```

Trial	1	2	3	4	5	6	7	8	9	10	11	12	13
Match?													
Trial	14	15	16	17	18	19	20						
Match?													

Based on the pooled data, what is the empirical probability that there'll be at least one match?

Lab 11. Probability Simulations: Poker Hands

In one version of Poker, each player gets a five-card hand of cards from a standard deck. Here are a few types of poker hands:

Full House – A hand consisting of 3 cards of the same denomination and 2 cards of the same denomination (i.e. 3 of a kind and a pair).
Pair– A hand consisting of exactly one pair of cards with the same denomination.

What is the probability of obtaining each of the above poker hands?

Again, we'll use Maple to simulate. The appropriate command this time is

```
>with(combinat,randcomb);
 randset := combinat[randcomb];
 cardDeck := {Ac,Ah,As,Ad, Kc,Kh,Ks,Kd,
Qc,Qh,Qs,Qd, Jc,Jh,Js,Jd, seq(i.c, i=2..10),
seq(i.h, i=2..10), seq(i.s, i=2..10), seq(i.d,
i=2..10)};
randset(cardDeck,5);
```

This returns the following:

$$\{Ah, Qd, 4\,d, 3\,s, 9\,c\}$$

where Ah stands for ace of hearts, Qd stands for Queen of diamonds, and so forth.

Simulate each game 20 times and keep track of your results in the following tables.

Full House

Trial	1	2	3	4	5	6	7	8	9	10
Full House?										
Trial	11	12	13	14	15	16	17	18	19	20
Full House?										

Total number of full houses _____

Pair

Trial	1	2	3	4	5	6	7	8	9	10
Straight?										
Trial	11	12	13	14	15	16	17	18	19	20
Straight?										

Total number of pairs _____

We will now pool the data. Based on the pooled data, the empirical probability of obtaining a full house is _____ and the empirical probability of obtaining a pair is _____.

Lab 12. Baseball Best-of-5 Series – Experimental Probabilities

The Baltimore Orioles and the Boston Red Sox are playing a best of 5 series. This means that whichever team is the first to win 3 games is the winner of the series. Let's first suppose that both teams are evenly matched. Before doing any calculations, use your intuition to answer the following questions:

What is the probability that the Orioles will sweep the Red Sox?

Is the series most likely to end in 3, 4, or 5 games?

On average, how long do you expect the series to last?

We'll use Maple to help us simulate this best-of-5 series. The script is as follows, where $n = 2$ and $k = 5$.

```
> randseries:=proc(n,k)
local i;
randomize();
RETURN(map(rand(1..n),[seq(i,i=1..k)]))
end proc;
```

Let Orioles := 1 and Red Sox :=2. Note that the series doesn't necessarily go to 5 games; if Boston sweeps the Orioles (i.e. any sequence that begins with 222) then the series is over in 3 games. Execute the procedure 20 times and record your results in the following table.

# of games needed	3	4	5
Orioles win the series			
Boston wins the series			

Pool your data with the rest of the class.

What is the experimental probability that the Orioles will sweep the series?
What is the experimental probability that Boston will sweep the series?
Is it most likely that the series will end in 3, 4, or 5 games?
What is the average number of games played?

Lab 13. Baseball – Binomial Probability

Assume that Ken Griffey, Jr. gets a hit with probability 1/3. What is the probability that in 100 plate appearances, Junior gets exactly 33 hits? Without doing any calculations, provide an educated guess.

The following procedure calculates the probability that Junior gets on base exactly k times in n attempts if he always gets a hit with probability p.

```
> with(combinat,numbcomb);
> prob:=proc(n,k,p)
      numbcomb(n,k)*p^k*(1-p)^(n-k)
  end proc;
```

Use the "prob" procedure to determine the probability that Junior gets on base exactly 33 times in 100 plate appearances. How close is the answer to your guess above? Reconcile any differences.

What is the probability that in 100 plate appearances Junior gets on base between 31 and 35 times? Use the following to calculate this probability.

```
> sumprob:=proc(n,p)
  sum('numbcomb(n,k)*p^k*(1-p)^(n-
k)','k'=31..35);
  end proc;
```

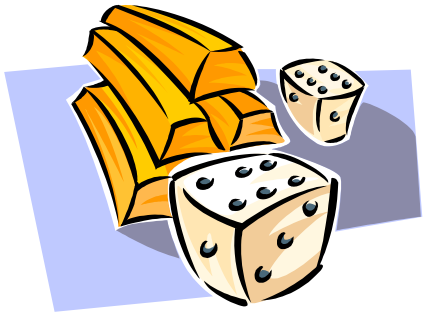
Modify the above script to answer the following questions.

What is the probability that in 100 plate appearances Junior gets on base between 27 and 37 times?

What is the probability that in 100 plate appearances Junior gets on base between 23 and 43 times?

What is the probability that in 100 plate appearances Junior gets on base between 13 and 53 times?

Lab 14. Expected Value Problems



Suppose five cards including an Ace are drawn from a standard deck. (If the five cards do not include an Ace, they are replaced and the deck is reshuffled.) What is the expected number of aces among the five cards?

We'll approach this problem using simulated data first. Begin by using Maple to randomly select 5 cards from a fair deck of 52.

The following script generates 50 5-card hands from a fair deck of 52.

```
>with(combinat, randcomb);
>carddeck:={Ac, Ac, As, Ad, Kc, Kh, Ks, Kd,
Qc, Qh, Qs, Qd, Jc,Jh,Js,Jd,seq(i.c,i=2..10),
seq(i.h,i=2..10),seq(i.s,i=2..10),
seq(i.d,i=2..10)};
>aces:={Ac,Ah,As,Ad}
>for i from 1 to 50 do
sort(randcomb(carddeck,5),lexorder);
end do;
```

Remember to eliminate hands that do not to contain at least one ace. What is the average number of aces based on your simulations?

Pool your data with four friends. What is the average number of aces based on your groups' data?

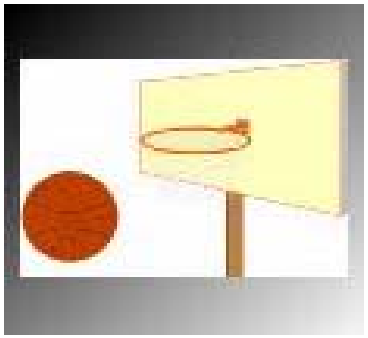
Suppose we were to simulate this experiment infinitely many times. What is your guess as to the average number of aces in a 5-card hand?

Now solve this problem using *theoretical probabilities*.

How close did your experimental guess come to the theoretical probability?

Lab 15. One and One

When a player is fouled in the game of basketball he or she sometimes gets what's called a "1-and-1 free-throw". This means if the player makes the shot, he/she scores a point and gets to shoot again. If the second basket is made, the player scores an additional point. On the other hand, if the player misses the first shot, he/she gets 0 points and does not get to try again.



1) Suppose that Carlos is a pretty good free-throw shooter – he's known to make 60% of the shots that he attempts**.**

   When Carlos goes to the line for a 1-and-1, is he most likely to score 0, 1, or 2 points? Explain.

2) Use Maple to simulate this problem. Carry out the simulation and determine which score Carlos is most likely to get in a 1-and-1 shooting situation.

Number of Points	0	1	2
Tally			
Total			

Is Carlos most likely to score 0, 1, or 2 points?

3) Use the simulation data in question 3 to find the *average* number of points that Carlos scored. In a sentence, give the practical significance of this number.

4) Based on your simulation data, what is the likelihood that Carlos will score 2 points? 1 point? 0 points? (These are the *experimental* probabilities.)

Outcome	0	1	2
Experimental Probability			

5) How do these results compare with what your answers to question 2?

6) Use a tree diagram to compute the *theoretical* probability that Carlos scores 0 points, 1 point, or 2 points in a 1-and-1 and then complete the chart below.

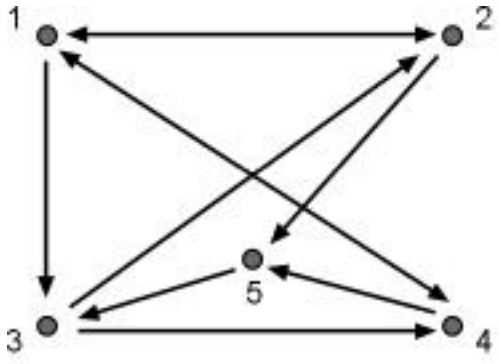
Outcome	Theoretical Probability
0	
1	
2	

7) Compare the *theoretical* probabilities with the *experimental* probabilities.

8) What is the expected number of points Carlos will score when he goes to the line?

Lab 16. Binary Relations: Influence

As part of its campaign to step up a recycling project, a group wants to entertain the most influential members of the city council. After some investigation, the firm's perception of pairwise influence among council members is as shown below. (An arc goes from member x to member y if x appears to influence y.)



Open the Maple file relations.mws, and select Execute Worksheet from the Edit menu. Save your worksheet using a different file name so that you do not write over this beginning file.
Define the above relation using the following command:

```
> R := [ [
[1,2],[1,3],[1,4],[2,1],[2,5],[3,2],[3,4],[4,1
], [4,5],[5,3] ], 5 ];
```

Compare the way that Maple draws the arrow diagram with the picture above.

```
> drawGraph(R);
```

Write the adjacency matrix *M* corresponding to this graph.

Check your answer above using the following Maple command:

```
> M := listToMatrix(R);
```

Suppose we want the matrix that shows *indirect* influence. Person x has **indirect influence** over person z if person x has direct influence over person y and person y has direct influence over person z. Use Maple to evaluate the matrix M^2, and explain what this matrix has to do with indirect influence.

```
> evalm(M^2);
```

What does the matrix M^3 mean in terms of the influence relation in this problem?

Now consider that matrix sum $M + M^2$. Compute this sum and explain how it relates to influence relation in this problem.

```
> evalm(M + M^2);
```