CALCULATED FICTION

MATH WORKSHOP 1

September 28, 2004

For this workshop you should work in a group with your project seminar.

PART I: ROAD-BUILDING

You have a plot of land in the shape of an equilateral triangle, as shown on the handout. (Equilateral means having three equal sides and three equal angles, each angle being 60°.)

You're trying to decide where to build a house on your land. The sides of the triangle are county roads, and you need to have access from the house to all three of them. Thus you have to build access roads that run from your house to each of the county roads.

Before you begin, choose one person to be your group's scribe. That person will be in charge of writing down the ideas your group has and the progress you make. This will free the rest of you up to focus on generating and exploring new ideas.

1. Where should you build the house to minimize the length of access roads that must be built? String will be provided so that you can work to find the answer experimentally. Don't go on until your group agrees on an answer. If you get stuck, ask Steven, Brian, or Kaiti for help.

2. How can you *prove* that your answer to # 1 is correct? Try to come up with an explanation that would convince any reasonable person that your answer is right. (Warning: This may be tricky.) Try out your ideas on the other members of your group; work with them to find the gaps in your reasoning.

Keep track of your work and your group's results; you should turn in your answers with your other work at the end of the workshop.

After about 20 minutes, we'll ask each group to share their results.

PART II: SETS & FUNCTIONS

The rest of today's workshop will introduce sets and functions, some of the basic building blocks of Mathematics. The concepts you learn in this workshop will provide useful vocabulary for exploring the interplay between Mathematics and language.

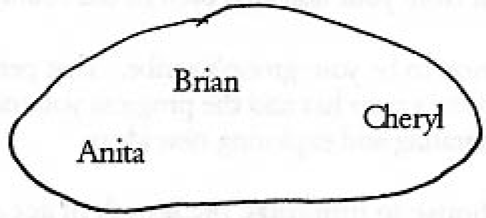
Sets and functions are simply more formal versions of things you're already familiar with. Sets are the math way of representing groups of things, and functions are the math way of representing processes. What follows is an attempt to make some of our intuitive ideas about these objects more precise.

Work through this packet with your project seminar group. You should progress through the workshop along with your group members; use them to develop your understanding. Check your work with each other. Ask for explanations if you don't understand, or help someone else if you do understand. Work together to make sure everyone gets the concepts being presented. And if you ever get stuck, ask for help! Steven and Brian and Kaiti are here to assist you.

A note about group work: Each of you needs to keep and submit your own sheets of work and answers. The group works together, but you need to have your own document to turn in and to later include in your portfolio.

SETS

What's a set? A set is just a collection of objects. We can draw a set as a blob with some things inside it:



We call the things inside the set the *elements* of the set. When the object x is an element of the set A, we write $x \in A$. When it's not, we write $x \notin A$.

We'd waste a lot of paper if we had to draw a blob every time we wanted to show a set, so it's a good idea to find different notation. Let's write a set down by putting a list of its elements in between some curly braces: {Brian, Cheryl, Anita}. If we want to give the set a name, we can use the equals sign to do so: S = {Brian, Cheryl, Anita}. That way we can refer to it later without having to recopy the whole thing. (Mathematical notation is often simply an attempt to save some writing.)

What can go into a set? For our purposes, we'll say that anything we can write down can be an element of a set. Hence these are all perfectly good sets:

{carrot, squash, rutabaga, turnip}
{Marie of Roumania, Abraham Lincoln, Beatrix Potter}
{Delaware}
{Supertramp, Friday, 7, catnip, velvet, long-haired chihuahua}

Notice that the elements of a set don't all have to be the same kind of thing.

ACTIVITY

1. Write down a few sets of your own. Play around with including different kinds of objects together in the same set. Try making a set with another set as an element; we can call them *nested sets*. Now make a set that has another nested set as an element. How deep can the nesting go?

2. Can you make a set with 0 elements? Can you make a set that has itself as an element?

What if we want to talk about a set with a lot of elements? The set of all 50 two-letter state abbreviations, for example:

{AL, AK, AZ, AR, CA, CO, CT, DE, FL, GA, HI, ID, IL, IN, IA, KS, KY, LA, ME, MD, MA, MI, MN, MS, MO, MT, NE, NV, NH, NJ, NM, NY, NC, ND, OH, OK, OR, PA, RI, SC, SD, TN, TX, UT, VT, VA, WA, WV, WI, WY}

That's dang annoying to write down. Let's write it in a shorter way: {AL, AK, ..., WI, WY}. Much faster; we let the "..." stand for the elements that continue the pattern. Using this notation, we can quickly write down some otherwise cumbersome sets:

{a, b, c, ..., z}
{Washington, Adams, Jefferson, ..., Clinton}
{1, 2, 3, ..., 1000000}

One note: we have to be careful with "..."; we must agree to only use it when there can be absolutely no doubt about what it replaces. For example, {1, 182, 16, ..., 5} doesn't make sense because we can't tell what the "..." stands for. If you use "...", be sure to start your set with enough elements to clearly indicate the pattern; three are usually sufficient.

We can also use "..." to write down some *infinite* sets. Consider {0, 1, 2, ...}. The pattern is clear: you just keep adding one. We aren't told where to stop, so we keep on going forever. The set {0, 1, 2, ...} comes up often enough that we give it a special name, N; it's the set of *natural numbers*.

Some more examples:

{a, aa, aaa, aaaa, ...}
{37, 36, 35, ...}
{1, 3, 5, ...} = the set of odd positive integers
{2, 4, 6, ...} = the set of even positive integers

ACTIVITY

3. Write down a few sets using "...". Make some that are finite and some that are infinite. Can you make a set this way that has two different kinds of things in it?

4. Come up with a reasonable definition of *infinite set*. Try to make your definition as rigorous as you can.

Yet another way to write down sets is using *set-builder notation*. For example, consider

{x : x is a former president of the United States}

Here we include all objects x that satisfy the specified condition. This is often the easiest way to write down a set. Some more examples:

{x : x is a two-letter abbreviation for a U.S. state}
{y : y is a word of English}
{w : w is a positive multiple of 5}
{z : z is a person who owns Brian's car}

This last set contains only one element, but it's still a perfectly good definition.

ACTIVITY

5. Write down a few sets using set-builder notation. Explore the capabilities of this notation. What's the largest set you can make? What's the most complicated set you can write down this way?

Note that the sets {Washington, Adams, Jefferson, ..., Clinton} and {x : x is a former president of the United States} have the same elements. Thus it makes sense to say that they are the same set.

This is a general rule: whenever two sets have exactly the same elements, we'll say that they're equal. That's our way of telling when two sets are the same. Maybe that sounds obvious, but it has an important consequence: it implies that it doesn't matter what order we write a set's elements in. For example, {Brian, Cheryl, Anita} = {Cheryl, Anita, Brian} since the two sets have the same elements.

ACTIVITY

6. Write down a set with at least six elements. Find four different ways to write down that same set, using each of the above types of notation at least once.

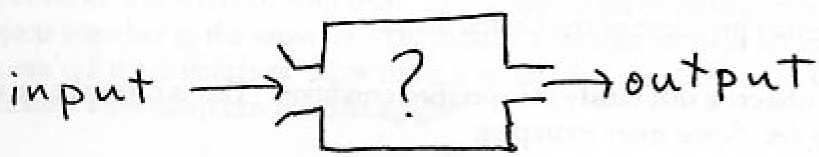
One more idea before we move on: if every element of a set A is also an element of the set B, then we say A is a *subset* of B, and we write $A \subset B$. For example, $\{h, q\} \subset \{a, b, c, ..., z\}$, since h and q are both elements of {a, b, c, ..., z}, but {Cheryl, Patty} $\not\subset$ {Brian, Cheryl, Anita}, since Patty is not an element of {Brian, Cheryl, Anita}. (Just as $\notin$ means "not $\in$", $\not\subset$ means "not $\not\subset$".)

ACTIVITY

7. Looking back over the sets you came up with for earlier questions, say which ones are subsets of which other ones.

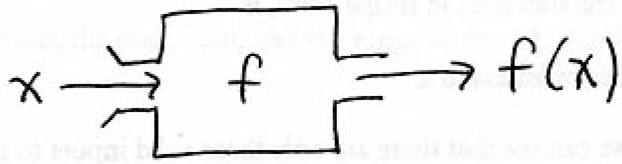
FUNCTIONS

We can think of a function as a machine. The machine takes in an object as input and produces an object as output:



As a first example, we can think of a vending machine; the input is a code (perhaps C3), and the output is a delicious snack (perhaps a baggie of Jerquee vegan jerky alternative).

Let's give the input a name; we'll usually call it x. Then if we call the function f, the machine diagram looks like this:



We tend to give functions names like f and g and h, or sometimes f_1 and f_2 and f_3 and ..., but really we can call them anything we like. Sometimes the name can remind us of what the function does. For example, if x is a sequence of characters, the function given by

reverse(x) = x in reverse order

is better off being called reverse than being called f or g.

The notation we use may look funny to you; f(x) looks like f TIMES x, right? But it doesn't mean that – it instead denotes the result of putting x into the f function machine. This takes a bit of getting used to.

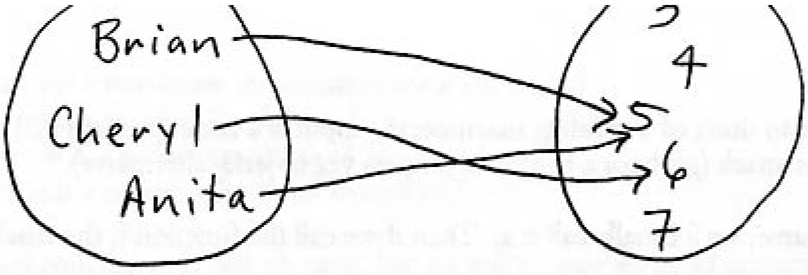
In general, we won't be too picky about how functions behave, but we will make one absolute requirement: for any one given input, the function must produce exactly one output. We want to avoid the sort of machine that has to decide which of two outputs to give, or that produces two outputs simultaneously; it may be nice when a vending machine does that, but we'll insist that anything we call a function must produce only one output for any given input. We also want to avoid machines that sometimes produce *no* output. Another way to say all of this is that functions must be *single-valued*.

We have a few different ways of defining functions. One good way is by giving a recipe that tells you what the function does to a particular input. Let's look at the reverse function as an example. What is reverse("abcde")? By the definition, it's "abcde" in reverse order, which is "edcba". Nothing fancy; you just take the input you're given and plug it into the definition.

ACTIVITY

8. Think of some examples of things that seem like functions. Are they single-valued? Write down some that turn things of one kind into things of another kind (e.g., codes into snacks), and write down some that turn things of one kind into things of that same kind (e.g., words into words).

Another good way to define a function is by drawing a diagram. We can put the input values in a set on the left and the output values in a set on the right. Then we can use arrows to tell us what the function does to each input value, as in this diagram:



Here we might say that the function, in recipe form, is

f(x) = the number of letters in x.

But from the diagram we can see that there are only three valid inputs to the function. So although the recipe would make sense in other circumstances, we allow the possibility that we might want to restrict a function to a limited set of inputs. We call the set of allowed inputs to a function the *domain* of that function. In this example, we can write

domain(f) = {Brian, Cheryl, Anita}

What about the set on the right? The set that contains all the possible output values is called the *codomain*. Here we can say

codomain(f) = {3, 4, 5, 6, 7}

We could have chosen lots of other codomains for this example; the only values we really need to have in there in this case are 5 and 6. Hence in the diagram above we could have put the set {1, 2, 3, 4, 5, 6, 7, 8} on the right-hand side, or {5, 6, 2999}, or even N.

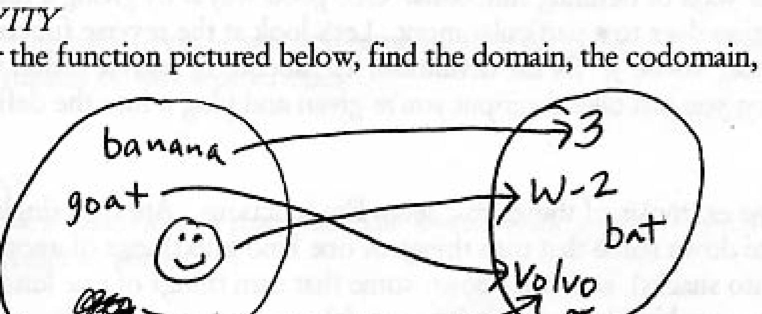
Now, 5 and 6 are special because they actually get used by the function f. For any function, we can form the set of outputs of the function and call it the *range*. In this example,

range(f) = {5, 6}

So the range is always a subset of the codomain, but they're not necessarily equal.

ACTIVITY

9. For the function pictured below, find the domain, the codomain, and the range.



For a function defined by a recipe, these sets may not be specified. (Think about the definition of reverse(x).) In that case, we'll say that the domain is the set of all inputs that make sense. For example, we might assume that domain(reverse) = {x : x is a sequence of characters}

since any sequence of characters can reasonably be reversed. In another setting, though, I might want to only talk about reversing sequences of letters, or sequences of numbers, or sequences of who knows what; in that case I can explicitly specify the domain.

ACTIVITY

10. Find the domain, the codomain, and the range of two of your functions from # 8.

PROJECT

Your project is due at the beginning of next week's math workshop. Each student must submit a project individually. On the first page of your submitted project, indicate who you worked with. (This time it should be your project seminar group, but it won't always be, and disclosing who you collaborated with is a good habit to get into.)

1. With your project seminar group or on your own, finish any workshop activities that you didn't get to during workshop time.

2. Find some other reading about sets or functions, and use it to learn an idea that isn't covered in this workshop packet. After you've understood the new idea, explain it in writing in your own words. (Good places to look for other reading are on the web and in the library; the point of this part is to get you looking for outside information.) Be sure to say what source you used.

3. Think about ways in which you can use the above ideas about sets and functions to model material from "The Library of Babel" and, time permitting, from the Oulipo Packet. How can you use sets and functions to describe structures or processes from the reading?

4. Get a scientific calculator and bring it to each math workshop from now on. Any calculator that says "scientific calculator" should suffice; any that doesn't probably won't suffice.