CALCULATED FICTION

WEEK 2 MATH HOMEWORK

September 27, 2007

The following is due next Tuesday at 10am in Math Workshop.

Reading

- This handout
- Pages 719-726 and 730-739 in *Mathematical Excursions* [ME]

Homework

- Do the problems embedded in the reading below.
- From section 11.1 in ME, do problems 2-18 (evens) and 27-30.
- From section 11.2 in ME, do problems 2-8 (evens), 12, 14, 16, 26, 28, 30, 38, 40, 44, 48, and 50.

SETS & FUNCTIONS

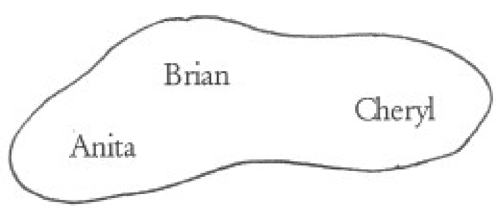
This reading will briefly introduce sets and functions, some of the basic building blocks of Mathematics. The concepts you learn here will provide useful vocabulary for exploring the interplay between Mathematics and language.

Sets and functions are simply more formal versions of things you're already familiar with. Sets are the math way of representing collections 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 everyday items more precise.

You may have noticed that ME has sections about sets and functions. It's a great idea to use that book as a resource as you read through this handout.

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* or *members* 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.

PROBLEMS

1. Write down a few sets of your own. Play around with including different kinds of objects together in the same set.

2. Note that a set is itself an object, so we can include sets as members of other sets. Make a set with another set as an element; we can call such things *nested sets*. Now make a set that has another nested set as an element. How deep can the nesting go?

3. Can you make a set with 0 elements? Write one down, or say why it's impossible to do so.

4. Can you make a set that has itself as an element? Write one down, or say why it's impossible to do so.

What if we want to talk about a set with a lot of elements? The set of all 50 two-letter U.S. 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, Bush}
{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 elements usually suffice.)

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

PROBLEMS

5. Write down a few sets using "...". Make some that are finite and some that are infinite.

6. Come up with a reasonable definition of *infinite set*. Try to make your definition as precise and 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}

That last set contains only one element, but this is still a perfectly good definition.

PROBLEMS

7. Write down a few sets using set-builder notation. Explore the capabilities of this notation.

8. What's the largest set you can write down using set-builder notation?

9. What's the most complicated set you can write down using set-builder notation? You'll have to figure out what "complicated" means to you in this context, so go ahead and write that down, too.

Note that the sets {Washington, Adams, Jefferson, ..., Clinton, Bush} and {x : x has been the 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.

PROBLEMS

10. 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 $A \subseteq B$. For example, $\{h, q\} \subseteq \{a, b, c, ..., z\}$, since h and q are both elements of $\{a, b, c, ..., z\}$, but {Cheryl, Patty} is not a subset of {Brian, Cheryl, Anita}, since Patty is not an element of {Brian, Cheryl, Anita}. When we want to write "is not a subset of", we can put a slash through the $\subseteq$ symbol, just as we put a slash through the $\in$ symbol to mean "not $\in$". (Unfortunately, $\subseteq$-with-a-slash-through-it doesn't seem to be an available symbol in Microsoft Word.)

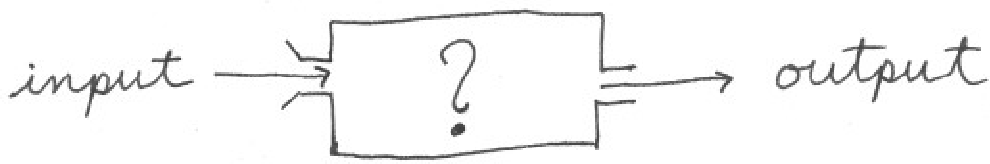
Additionally, if $A \subseteq B$ and also $A \neq B$, then we say A is a *proper subset* of B, and in that case we write $A \subset B$. If A is not a proper subset of B, we write $A \not\subset B$. (Yep – Microsoft Word has $\not\subset$ but not the other one.) So, for example, $\{h, q\} \subset \{a, b, c, ..., z\}$, but $\{h, q\} \not\subset \{h,q\}$ since the two sets are equal.

PROBLEMS

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

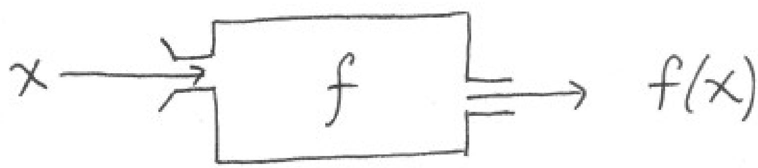
FUNCTIONS

We can think of a mathematical 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 a little funny; $f(x)$ 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 one and only 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 exactly one output for any given input. However, we *are* allowed to restrict which kinds of objects go into any given function; think of it as a filter that we can put over the input hole. So we can talk about the *reverse* function without having to worry about what *reverse* would do to, say, your digestive system. 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*.

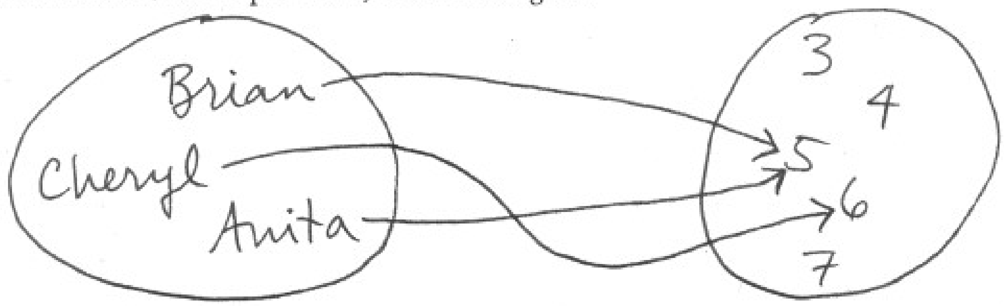
We have a few different ways of defining functions. One good way is by giving a recipe that tells you what the function does to a particular input. Let's look at the *reverse* function as an example. What is *reverse*("abcde")? By the definition, it's "abcde" in reverse order, which is "edcba". Nothing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

PROBLEMS

12. Think of some examples of things that seem like functions. Are they single-valued? What restrictions do you have to make on the kinds of things that can be inputs?

13. Write down some functions 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 the name of 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 more limited set of inputs than is strictly necessary. 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 *range*. Here we can say

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

We could have chosen lots of other ranges 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**.

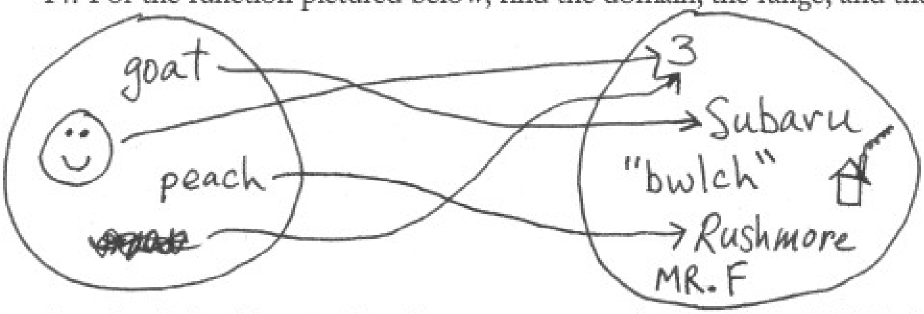
Now, 5 and 6 are special because they actually get used by the function f. For any function, we can form the set of outputs of the function and call it the *image*. In this example,

image(*f*) = {5, 6}

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

PROBLEMS

14. For the function pictured below, find the domain, the range, and the image.



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. (Note that we just used one function as an input to another function. Snazzy, huh?) 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.

PROBLEMS

15. Find the domain, the range, and the image of two of your functions from problem 13.