LARSON—MATH 353–CLASSROOM WORKSHEET 16
Perfect Numbers—Files.

1. Log in to CoCalc.
   (a) Start the Chrome browser.
   (b) Go to `https://cocalc.com`
   (c) Login (**your VCU email address** is probably your username).
   (d) You should see an existing Project for our class. Click on that.
   (e) Click "New", then "Worksheets", then call it **c16**.

Perfect Numbers

A number (integer) n is *perfect* if the sum of its proper divisors (the divisors less than n) equals n (or equivalently that the sum of all divisors equals $2n$). 6 is the smallest perfect number.

2. Finding Examples. Sage will produce the divisors of a number. Run: `6.divisors()`.
   Here is code to test if an integer n is perfect:

```
def is_perfect(n):
    if sum(n.divisors())==2*n:
        return True
    else:
        return False
```

   Use this to find as many perfect numbers as you can.

3. For each perfect number n factor it (using `factor(n)`) and see if you can conjecture anything about these examples. Is there a pattern?

4. If you have a conjecture, can you think of any *reasons* why it might be true?

5. What can we say about the existence of *odd* perfect numbers?

Files

6. Now it is the case on any larger program that you will want to use functions you have previously defined. These are called *tools*. Instead of copying and pasting from your old code. You can save them as *files* and load them as needed.
   (a) Click "New". Type `heads_from_n_flips.sage` and then click "file".
   (b) Define the function:

```
def heads_from_n_flips(n):
    heads=0
    for i in [1..n]:
        if random()<.5:
            heads=heads+1
    return heads
```

(c) Click "Save" and then go back to your **c16** worksheet.

(d) Type `load("heads_from_n_flips.sage")` and evaluate.

(e) Now try `heads_from_n_flips(100)` a few times. You never need to write this function again. You have a tool!

Reading in, and working with, data files is an important ability. First we will create a data file. Then we will read it in line-by-line, and then we will work with the data.

An important thing to know/note is that a file is actually a big *string*. You can read the lines of a file with `readline()`. Those lines are also strings (and not numbers - despite how they look). If you want numbers they must be converted to numbers.

7. (a) Go to: `http://projecteuler.net/problem=13`. Copy the one hundred 50-digit numbers there.

   (b) Click "New", type in `one_hundred_numbers.txt` as the name of your file, then click "File".

   (c) Paste in your numbers and "Save".

   (d) Now go back to your **c16** worksheet.

   (e) Type in:

```
data=open("one_hundred_numbers.txt")
numbers=[]
number_string=data.readline()
while(number_string!=""):
    number=Integer(number_string)
    numbers.append(number)
    number_string=data.readline()
```

   You have a *list* of numbers. You can use built-in Sage functions to find out statistics about this list. How many numbers are there? What is the biggest number? What is the sum of these numbers? What is the average of these numbers? What is their median?

8. Now we will create a file `testio.txt` in *write* mode (hence the "w"), and write something to it. The *close* command forces the writing to happen and flushes the *buffer*. Now you can't write anything else to the file without reopening it.

```
datafile=open("testio.txt","w")
datafile.write("hello world!")
datafile.close()
```

9. Go to Files, find testio.txt and click on it to see what's in there.

10. **Getting your classwork recorded**

    When you are done, before you leave class...

    (a) Click the "Make pdf" (Adobe symbol) icon and make a pdf of this worksheet. (If Cocalc hangs, click the printer icon, then "Open", then print or make a pdf using your browser).

    (b) Send me an email with an informative header like "Math 353—c16 worksheet attached" (so that it will be properly recorded).