In[23]:=

Our First Mathematica Notebook

Out[23]=

```
First Mathematica Notebook Our
```

Many topics in number theory are best learned using Wolfram Mathematica or a comparable computer algebra system such as SageMath or Mathematica. (Some, but not all, of these tasks can be formed on your calculator or Matlab.) Some of our homework assignments will assume you have access to such computational resources either

1. Wolfram Alpha, accessed through a browser;
2. Mathematica, at a UW campus workstation;
3. Mathematica, from off campus, logged in remotely to a UW computer lab;
4. Mathematica, installed on your computer using a student license provided by UW; or
5. SageMath or other equivalent software.

See https://uwyo.teamdynamix.com/TDClient/1940/Portal/Requests/ServiceDet?ID=9596 for help with accessing Wolfram Software.

Here we demonstrate the use of a Mathematica notebook, which allows us to organize, save and share our work in a formatted document, using headings and text documentation in addition to Mathematica code. All Mathematica documents that I show you in our class, including this one, will be shared with you so that you can imitate and re-use these code segments in your own study and homework solutions. Let's demonstrate the use of Mathematica through a series of computational examples.
To compute the algebraic expression shown, enter as "code", then shift-enter:

In[24]:=

```
3*(1+17^11)
```

Out[24]=

```
102 815 688 922 902
```

In[25]:=

```
3/4 + 11/13
```

Out[25]=

$$\frac{83}{52}$$

Some computations use specialized commands whose arguments are enclosed in brackets:

In[26]:=
```
Factorial[22]
```

Out[26]=

1 124 000 727 777 607 680 000

Solve a quadratic equation for x. Omit the “Clear[x];” command if the variable “x” hasn’t yet been assigned.

In[27]:=
```
Solve[x^2+13*x+2==0, x]
```

Solve: 163 is not a valid variable.

Out[27]=

Solve[False, 163]

Sometimes an attempted computation like this fails because your variable name (in this case x) has already been assigned a value. One remedy is to first clear the variable name, and try again:

In[28]:=
```
Clear[x]; Solve[x^2+13*x+2==0, x]
```

Out[28]=

$$\left\{\left\{x \to \frac{1}{2}\left(-13 - \sqrt{161}\right)\right\}, \left\{x \to \frac{1}{2}\left(-13 + \sqrt{161}\right)\right\}\right\}$$

If you don’t know the Mathematica command to compute something, use instead “NaturalLanguageInput”, which will first try to translate your command into Mathematica code. Mathematica will try to interpret your natural language as a command (which you should first verify) and then proceed to give the answer (when you click on the boxed equation symbol “=” on the left):

solve x^2+13x+2=0 for x

In[ ]:=
```
Solve[x^2 + 13 * x + 2 == 0, x]
```

Out[29]=

$$\left\{\left\{x \to \frac{1}{2}\left(-13 - \sqrt{161}\right)\right\}, \left\{x \to \frac{1}{2}\left(-13 + \sqrt{161}\right)\right\}\right\}$$

It will be your responsibility to check that Mathematica has correctly interpreted your “natural language” (it doesn’t always!). Once you have learned the correct Mathematica code, you can use that directly.

We will often perform operations using modular arithmetic. This example uses modular arithmetic:

7^13 mod 11

In[ ]:=
```
Mod[7^13, 11]
```

Out[30]=

2

This example asks us to solve a congruence. We apparently need to click on the equation symbol (highlighted equals) to compute:

solve x^13=2 mod 11 for x

In[ ]:=
```
Solve[x^13 == 2, x, Modulus → 11]
```

Out[31]=

{{x → 7}}

Separate multiple commands using semicolons. Commands ending in a semicolon will not result in displayed output; only the last command (without a semicolon) will be shown:

In[32]:=
```
a=2^5+1; b=2^a+1; c=b^2+1
```

Out[32]=

73 786 976 312 018 075 650

For use in class demonstration, I may prefer a different approach which displays all values of **a,b,c**, for example:

In[33]:=
```
a=2^5+1; b=2^a+1; c=b^2+1; Print["(a,b,c) = (",a,",",b,",",c,")"]
```

(a,b,c) = (33,8 589 934 593,73 786 976 312 018 075 650)

In some demonstrations, I may want to show the full algebraic expression leading to the final value:

In[34]:=
```
Print[HoldForm[(2^HoldForm[2^5+1]+1)^2+1]," = ",c]
```

$\left(2^{2^5+1}+1\right)^2+1$ = 73 786 976 312 018 075 650

Although these practices are helpful when making class demonstrations, I am not expecting you to adopt any of these advanced features when using Mathematica for your own purposes! A simpler choice of syntax for this was shown above; or instead, use something like:

In[35]:=
```
(2^(2^5+1)+1)^2+1
```

Out[35]=

73 786 976 312 018 075 650

Also, “%” represents “the previous expression”; so you could show all intermediate steps by avoiding semicolons, and ending each line with “enter” (NOT “shift-enter”):

In[36]:=
```
2^5+1
2^%+1
%^2+1
```

Out[36]=
33

Out[37]=
8 589 934 593

Out[38]=
73 786 976 312 018 075 650

Mathematica is a very extensive computer algebra system, most of which we will not see or use in this course. Let’s however look at some typical computations we will be using, which I will introduce using the natural language feature:

list all the prime numbers less than 100

In[ ]:=
```
Prime[Range[PrimePi[99]]]
```

Out[39]=
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}

the next prime number after 10^6

In[ ]:=
```
NextPrime[10^6]
```

Out[40]=
1 000 003

the greatest common divisor of 399 and 567

In[ ]:=
```
GCD[399, 567]
```

Out[41]=
21

inverse of 47 mod 100

In[ ]:=
```
PowerMod[47, -1, 100]
```

Out[42]=
83

how many primes are there between 1000 and 1000000

In[ ]:=
```
PrimePi[1 000 000] - PrimePi[1000]
```

Out[43]=
```
78 330
```

Let’s try to list all primes between a googol and a googol+1000 (there can’t be very many in this interval). Mathematica’s natural language feature tries to do this the wrong way and fails:

list all primes between 10^100 and 10^100+1000

In[ ]:=
```
Prime[Range[PrimePi[Ceiling[
      10 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 ⋱
       000 000 000 000 000] - 1] + 1, PrimePi[
   10 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 ⋱
    000 000 000 100]]]
```

Range: Range specification in

Range[1 + PrimePi[999999999999999999999999999999999999999999999999999999999999999999999999999999999999⋱
999999999999999], PrimePi[1000000000000000000000000000000000000000000000000000000000000000000000⋱
0000000000000000000000000000100]] does not have appropriate bounds.

Out[44]=
```
Prime[Range[1 + PrimePi[
    9 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 ⋱
     999 999 999 999 999 999 999 999 999], PrimePi[
   10 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 ⋱
    000 000 000 000 000 000 000 000 100]]]
```

Instead, we test each integer in the required range, and individually test each one for primality. This gives us the complete list of primes in the desired interval (just two of them). We can use a “for loop”:

In[45]:=
```
googol=10^100; For[n=googol, n≤googol+1000, n++, If[PrimeQ[n], Print[n," is prime"]]]
```

```
10 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 -
  000 000 000 000 267 is prime
10 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 -
  000 000 000 000 949 is prime
```

Similarly, let’s use a “while loop” to list all prime numbers less than 1000 of the form $n^2 + 1$:

In[46]:=
```
n=0;
While[n^2<1000, If[PrimeQ[n^2+1], Print[n^HoldForm[2]+1," = ",n^2+1," is prime"]]; n++]
```

$2 = 2$ is prime

$1 + 2^2 = 5$ is prime

$1 + 4^2 = 17$ is prime

$1 + 6^2 = 37$ is prime

$1 + 10^2 = 101$ is prime

$1 + 14^2 = 197$ is prime

$1 + 16^2 = 257$ is prime

$1 + 20^2 = 401$ is prime

$1 + 24^2 = 577$ is prime

$1 + 26^2 = 677$ is prime

Next, let's give an example of a procedure (i.e. function or subroutine) in Mathematica, which accepts one positive integer n as input (note the underscore in specifying this variable as input, i.e. **n_**). Its output is a list the ways that n can be expressed as a sum of two integer squares. In order to suppress duplicates, we will ask for pairs (x, y) such that $n = x^2 + y^2$ where $0 \leq x \leq y$. It is important to realize that this naive code only works for "small" values of n. Later in the course we will see how to express a large integer n (possibly hundreds of digits long) as a sum of two squares, whenever such an expression exists (and why we would care about doing so).

```
In[48]:= sum2squares[n_] := (x=0; y=Sqrt[n];
           While[x≤y, y=Sqrt[n-x^2]; If[IntegerQ[y], Print[x^HoldForm[2]+y^HoldForm[2]," = ",x^2+y^2]]; x
```

```
In[49]:= sum2squares[109]
```

$3^2 + 10^2 = 109$

```
In[50]:= sum2squares[49]
```

$0^2 + 7^2 = 49$

```
In[51]:= sum2squares[52200]
```

$54^2 + 222^2 = 52\,200$

$90^2 + 210^2 = 52\,200$

$114^2 + 198^2 = 52\,200$

```
In[52]:= sum2squares[52201]
```

$45^2 + 224^2 = 52\,201$

```
In[53]:= sum2squares[52202]
```

Use a "for loop" to write each $n \in \{52\,200, 52\,201, \ldots, 52\,210\}$ as a sum of two squares whenever possible:

In[54]:=
```
For[n=52200, n≤52210, Print["Write ",n," as a sum of two squares"]; sum2squares[n]; n++]
```

```
Write 52 200 as a sum of two squares
```

$54^2 + 222^2 = 52\,200$

$90^2 + 210^2 = 52\,200$

$114^2 + 198^2 = 52\,200$

```
Write 52 201 as a sum of two squares
```

$45^2 + 224^2 = 52\,201$

```
Write 52 202 as a sum of two squares
Write 52 203 as a sum of two squares
Write 52 204 as a sum of two squares
Write 52 205 as a sum of two squares
```

$26^2 + 227^2 = 52\,205$

$58^2 + 221^2 = 52\,205$

$142^2 + 179^2 = 52\,205$

$157^2 + 166^2 = 52\,205$

```
Write 52 206 as a sum of two squares
Write 52 207 as a sum of two squares
Write 52 208 as a sum of two squares
Write 52 209 as a sum of two squares
```

$15^2 + 228^2 = 52\,209$

```
Write 52 210 as a sum of two squares
```