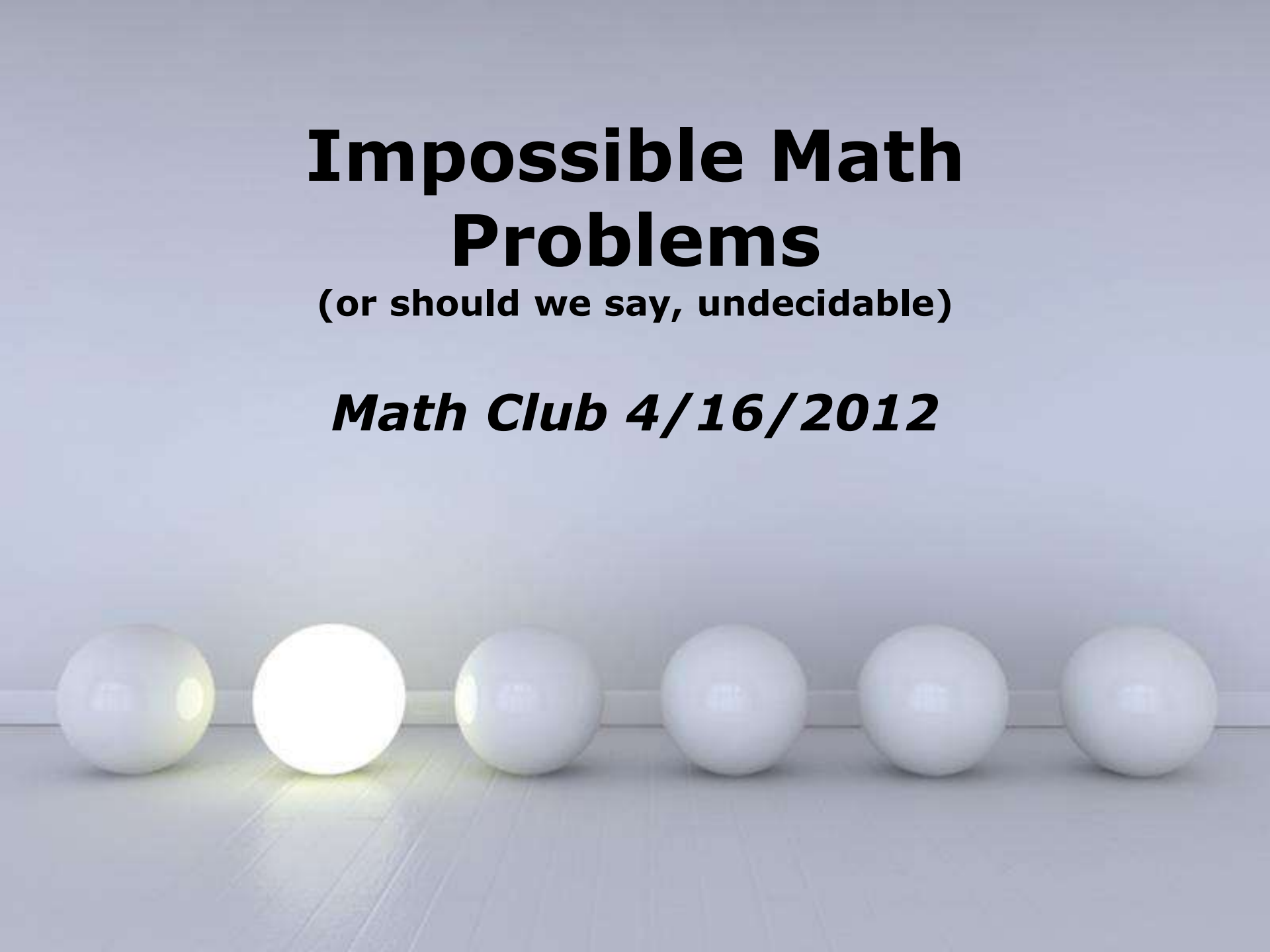
Impossible Math Problems

(or should we say, undecidable)

Math Club 4/16/2012



Goldbach's Conjecture (unsolved)

Take any even number n.
Then n is a sum of two prime numbers.

For instance, 20 = 3 + 17.

Verifying Goldbach's

Suppose we want to make sure that Goldbach's conjecture is true for some even n.

How can we do this?

There are finitely many ways to write n as a sum of two odd numbers, say p, q.

So we can check all of them.

Goldbach Version 2.0

Take some even number n.

We want to confirm that there exists two primes with difference n.

For instance, if we pick $n = 20$, then 53 and 73 work as our two primes.

Verifying Goldbach 2.0

How can we confirm this?

Start with a list of primes: 2,3,5,7,11,13…

Add n to each of them: 22, 23, 25, 27, …

Go through this list until you find a prime!

But honey, will it stop?

Let $g_+(n)$ return 1 if n can be written as a sum of two primes, otherwise 0.

Let $g_-(n)$ return 1 if n can be written as a difference of two primes, otherwise 0.

Now, a $g_+(n)$ calculation will always stop and give 1 or 0.

But what about $g_-(n)$?

Goldbach 2.0 (might be) undecidable!

Two cases:

1. **There does exist some primes so that $p - q = n$. The computation will find these, and stop.**

2. **There doesn't exist any of these prime pairs. Unfortunately, the computation can never prove that it doesn't exist.**

So if it's case 2, our program runs forever!

‘might be’?

Is $g_{-}(n)$ really undecidable?

Maybe not. Perhaps a member of the HWW math club can do some math wizardry and come up with a finite algorithm to do it.

An actual undecidable problem

Here's a program that quickly stops:

```
For every n from 1 to 100:
        print n
```

Here's one that doesn't:

10: print ":3"
20: go to 10

Does it stop?

Wouldn't it be useful to have a function that takes any program and decides if it will stop?

```
function stops(program, input):
        if(some regex wizardry):
                return YES
        else:   return NO
```

Then we never have to wait around and wonder if our program will stop!

An unusual program

In order for this to work, this function has to produce an answer for every possible input.

Suppose an alien programmer came up with this unusual program:

```
function alien(program):
        if stops(program, program):
                enter infinite loop
        otherwise exit and stop.
```

The paradox

What if we run alien() **on itself? Will it stop or will it run forever?**

Ideally, stops(alien,alien) **should tell us.**

- **If** stops(alien,alien) **returns true, then the same function enters an infinite loop, so** stops() **is wrong!**
- **If** stops(alien,alien) **returns false, then the function does stop, so again** stops() **is wrong!**

Conclusion

Here's our function for determining if a program will stop:

```
function stops(program, input):
        if(some regex wizardry):
                return YES
        else:   return NO
```

Problem is, if you give it the alien() **function, it cannot logically give the right answer!**

Ergo, this function cannot exist.