25
CONJECTURES
OF A 15 YEAR OLD :)

—RAGHAVENDRA N BHAT,
X C,
MADHAVA KRIPA SCHOOL,
MANIPAL,
INDIA

Foreword

This book is the collection of twenty five conjectures that I discovered. For those of you who don't know it, a ‘Conjecture’ is basically like a theorem. However, it can be stated as a theorem which has not been proved, yet *seems* to be true; meaning, it has been tried out on many numbers, and it hasn't yet been disproved. But, a theorem is generally something which has been proved. So, we can say that a conjecture is a ‘statement’ that remains to be proven or disproven.

This short book contains twenty five conjectures which I discovered. It so happens that all these conjectures are related to prime numbers, a field which I found myself fascinated by, right from the age of 8. However, it was during my 8th grade, that I actually started working out on primes and trying out ideas and calculations on them. I did come up with vague ideas on primes and some kind of conjectures, you can say. But it wasn't until my 10th grade that I was actually able to put these into action; because in grade 10, i tried trying out these ideas using computer programming.

When I used computer programming to prove these ideas and conjectures, I could see, for instance:
a: How far did they work ?
b: For what kind of numbers did it work ?
c: Why didn't it work for some ?
And some other such questions.

After all, for a conjecture to be true, there shouldn't be exceptions, or cliches. So, it had to be tried out on a fair number of numbers, before they could be called, a ‘Conjecture’.

It was on one fine day (night actually), that I came up with something valuable for the first time. A math teacher in our school, Mrs. Mala ma’am, had given be a book, from a near by library; after she saw a presentation given by me to some teachers and students. She thought the book would interest me. The book had a lot of tough math in it,

which, unfortunately I couldn't quite understand. So, I looked at the index and straightaway jumped to the chapter on primes.

I was casually reading the book. My mind began wandering (between primes, cricket, and other things). But maybe, due to some good fortune, it stuck to primes for a tiny bit longer. I thought of the well knows theorem, that every number could be written in the form of a product of primes. It fascinated me, and I began to think : Okay, we are talking about multiplication. Why not something simpler, like addition ?

I wondered: was there a way to write numbers, as a ‘sum’ of primes, rather than as a ‘product’ of primes ? Then it struck me. I tried out a few examples. 10 was 3 + 7, 20 was 3 + 17, 30 was 7 + 23. I just went on trying this for different even numbers, and it always worked. I shouted “Every even number can be written as a sum of two primes”. My dad, Dr. Narasimha Bhat, was nearby, reading something. He heard me. I repeated my thought, “Every even number can be written as a sum of two primes”. I myself, to be frank, didn't know how this could be possible.

Dad was quite fascinated, and suggested me to write a program on it and see how far it worked. So the two of us went downstairs to the computer. My dad helped me write this first program. I had learnt basic programming in my 8th std. holidays, guided by my uncle Dr. Prashanth Bhat and an online course on edX.org.

It took some time to get those skills back, but soon the program was ready. It was 11:30 at night, but it didn't matter. This was something special. Something told me it was special. So we ran the program.

And guess what !! The program went on and on and on. Every single even number it tried, seemed to simply bow to this theory, easily giving two primes which would add up to it. It went till 100,000. So far, for every even number, right from 2 till 1 lakh, could be written as a sum of two primes.

Then I got a crazy thought. It was something like a curtain to block sunlight. What if it was already discovered? That would be like pricking a balloon with a pin. Me, being crazy as usual, suggested this to my dad. “Shall we check if this is already discovered ?”. We

stopped the program and searched in google, ‘Even number as a sum of primes’.

And guess what !! It was already discovered. :(. That was a little harsh on my side actually. It turned out that the guy who discovered this thing was Goldbach, and this was actually called the Goldbach conjecture. This was discovered way back in the the 19th century, and Goldbach is pretty much knows for this conjecture. It still remains a conjecture actually. It has not yet been proved. But, it seems to work fine for millions and trillions of numbers.

The most powerful computers in the world are trying their best to disprove this conjecture, but to no avail. So, yes! Every even number in the universe can be written as a sum of two primes.

I felt sad that it was already discovered. Yet, there was that tiny bit of pride and satisfaction on discovering something that great, all by myself, that too without any pain-stacking effort. I thanked the book and Mala ma’am in my head.

That was the story of the first prime related idea that came to me, which unfortunately was already known. But, in this book, you shall see twenty five more, similar (but some slightly more complex ;)) conjectures, which as far as I know, have not yet been discovered. I have also attached the Python codes after each conjecture. So, those of you who know computer programming, can run these files to get results for large numbers.

This book is a humble effort from my side to put my conjectures in print. The titles are all related to cricket, because that is something I love too.

The book is written with the motive that all my friends and piers all around the world, will have the motivation to discover more such conjectures, or theorems, because, after all, we are one, and I am not much different from any of you. So if I can, so can you ! :).

Conjecture 1 — The odd sixer

Conjecture — Every odd number greater than 7, can be written as a multiple of six, plus a prime.

Quite simple actually; straight forward. Every odd number can be written in the form a + b, where ‘a’ is a multiple of six, and ‘b’ is any prime number.

For convenience, we start after 7, so as to avoid the number ‘1’. Here are some examples:-

1. 9 = 6 + 3
2. 11 = 6 + 5
3. 13 = 6 + 7

and so on. (something big — 5109 = 5106 + 3; something very big, 1000027 = 24 + 1000003.)

The code is quite simple too. Don't get freaked out. If you know programming, its just a piece of cake.

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=9
```

```
b=6
while True:
    if primetest(a-b)==True and (a-b)>1:
        print(str(a) + ' = ' + str(b)+ ' + ' + str(a-b))
        #print(str(a))
        a+=2
        b=6
    else:
        b+=6
        if b>a:
            break
```

Copy paste it to python and run it to get all the results.

Conjecture 2 — The square cut 6

Conjecture — Every multiple of 6 can be written as a + p1, where ‘a’ is a square number and p1 is a prime (or 0).

Again, quite simple; straight forward. Every number which is a multiple of 6 can be written in the form a + b, where ‘a’ is a square number, and ‘b’ is any prime number, or 0.

The ‘0’ comes in when, the multiple of 6 itself might be a square number !. Eg. 36 = 36 + 0. Anyway, this doesn't usually happen. So lets look at the examples. Remember, the first number cant just be any number; it has to be ‘square’ of a natural number.

Here are some examples:-

1. 6 = 1 + 5
2. 12 = 1 + 11
3. 96 = 25 + 71

(something big — 6414 = 25 + 6389; something very big, 602208 = 121 + 602087.)

As the multiples increase, the observation here is that, the first number doesn't actually enlarge drastically. It remains quite small. But the primes become quite huge.

For example:

18001128 = 361 + 18000767. 18000767 turns out to be a prime. However, the first number is just 19 squared !

Computer code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=6
b=1
correct=0
wrong=1
while True:
    c=a-(b*b)
    if primetest(c)==True:
        print(str(a) + ' = ' + str(b**2) + '+' + str(c))
        correct+=1
```

```
        a+=6
        b=1
    else:
        b+=1
        if b**2>a:
            break
```

Conjecture 3 — The multiples of the hat trick 6 (18)

Conjecture — Every multiple of 18 has a prime number P to it, such that P can be added or subtracted from the multiple to get two primes p1 and p2.

This one is a double blow.
It is capable of producing two primes; that too with the help of a third prime. So, its actually a triple blow. Every multiple of 18 has a prime number to it, which can be added to the multiple, to obtain a prime; or can be subtracted fem the multiple to get another prime.

So, if we take the multiple of 18 as ‘x’, and the prime number as P, we can say that:

1. x + P is a prime
2. x - P is also a prime

And, P is same. So with just one prime number, we can generate two more.
Some examples :-

1. 18 —5, 18 + 5 = 23, 18 - 5 = 13
2. 126 —13, 126 + 13 = 139, 126 - 13 = 113

(something big — 18270 —17, 18270 + 17 = 18287, 18270 - 17 = 18253)

(something very big — 12600486 —163, 12600486 + 163 = 12600649, 12600486 - 163 = 12600323)

As the multiples increase, the observation here is that, the prime number that is added or subtracted doesn't become quite huge.The prime numbers formed by adding and subtracting are quite large.

Code time —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=18
b=3
while True:
    if primetest(a-b)==True and primetest(a+b)==True:
        print(str(a) + ' + ' + str(b)+ '= ' + str(a+b))
        print(str(a) + ' - ' + str(b)+ '= ' + str(a-b))
        print(str(' '))
        a+=18
        b=3
    else:
        b+=2
        while primetest(b)==False:
            b+=2
        if b>a:
            break
```

Conjecture 4 — Multiple runs after the half century (part 1)

Conjecture — For every even number x, greater than 50 there exists a prime number P smaller than its half, such that x * P + 1 is prime.

Now, we begin to bring in multiplication. Again, we generalize to suit every even number. To rule out exceptions, we being after the number fifty.
The conjecture says that, there exists a prime number, which is smaller than the even number, actually smaller than its half; such that when we multiply the even number by the prime number and add 1, we get another, large, prime.

Eg:

1. 50 * 3 + 1 = 151
2. 100 * 7 + 1 = 701
3. 172 * 13 + 1 = 2237

(something big — 5164 * 3 + 1 = 15493)
(something very big —800906 * 11 + 1 = 8809967)

Again, the prime is found to be quite small, yet the prime formed is very large.

Here comes the code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
```

```
            return False
        else:
            y+=1
    if z!=1:
        return True

a=50
b=3
while True:
    if primetest((a*b) + 1)==True:
        print(str(a) + ' * ' + str(b) + ' + 1 = ' + str((a*b + 1)))
        #print(str(a))
        a+=2
        b=3
    else:
        b+=2
        while primetest(b)==False:
            b+=2
        if b>a/2:
            break
```

Conjecture 5 — Multiple runs after the half century (part 2)

Conjecture — For every even number x, there exists a prime number P smaller than its half, such that x * P - 1 is prime.

Almost similar to Conjecture 4. Here, instead of adding 1, we are subtracting one. Everything else is similar.
Examples :

1. 50 * 3 - 1 = 149
2. 62 * 7 - 1 = 433
3. 74 * 31 - 1 = 2293

(something big — 5136 * 5 - 1 = 25679)
(something very big —900128 * 79 - 1 = 71110111)

Code is also similar.

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=50
b=3
while True:
    if primetest((a*b) - 1)==True:
        print(str(a) + ' * ' + str(b) + ' - 1 = ' + str((a*b - 1)))
        #print(str(a), str(b))
        #print(str(a))
        a+=2
        b=3
    else:
        b+=2
        while primetest(b)==False:
            b+=2
        if b>a/2:
            break
```

Conjecture 6 — 10000 in tests

Conjecture — Every multiple of 10000 can be written as a + p1, where a is a cube number and p1 is a prime. (or 0)

This conjecture deals with a stunning property of multiples of 10000. Every one of those multiples can be written as a sum of a cube number and a prime ! We talked about square numbers, now cubes.

The cube number tends to be a lot smaller than the prime.
Some examples —

1. 10000 = 27 + 9973
2. 80000 = 1 + 79999
3. 170000 = 343 + 169657

(something big — 3210000 = 2197 + 3207803)
(something very big —101970000 = 343+101969657)

So, cubes can be used to create terrifyingly huge primes, quite in the vicinity of multiples of 10000.

Here's the code on Python:-

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True
```

```
a=100000000
b=1
correct=0
wrong=1
while True:
    c=a-(b*b*b)
    if primetest(c)==True:
        print(str(a) + ' = ' + str(b**3) + '+' + str(c))
        correct+=1
        a+=10000
        b=1
    else:
        b+=1
        if b**3>a:
            break
```

Conjecture 7 — 10000 whole with square cuts

Conjecture — For every number a, greater than 10000, there exists a number b, smaller than its square root such that, (a+b)^2 + 1 is prime.

Now, we bring in (a + b)^2. We refer to the fact that, for any number, (even or odd), greater than 10000, there exists a smaller number, (smaller than the number's square root), such that when we add the two numbers, square them and add 1, we get a prime.

Slightly complicated, but unbelievably interesting.

Examples will make it less confusing.
1. 10000 — (10000 + 6)^ 2 + 1 = 100120037
2. 10001 — (10001 + 5)^ 2 + 1 = 100120037

3. 10002 — (10002 + 4)^ 2 + 1 = 100120037

(something big — 45010 — (45010 + 24)^ 2 + 1 = 2028061157)
(something very big — 4056008 — (4056008 + 32)^ 2 + 1 = 16451460481601)

Once more, the prime numbers formed are very large compared to the numbers used to get them.

Code — prints the number after 10000, the small number to be added, the prime.

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=10000
b=2
while True:
    if primetest(((a+b)**2)+ 1)==True:
        print(str(a), str(b), str((a+b)**2 + 1))
        a+=1
        b=2
    else:
        b+=1
        if b**2>a:
            break
```

Conjecture 8 — 10000 whole with triple centuries

Conjecture — For every number a, greater than 10000, there exists a number b, smaller than its square root such that, (a+b)^3 + 3 is prime

We continue our quest with 10000. Now, we turn to cubes again. This is similar to the previous conjecture. But it states that there exists a number less than the square root which can be added, and the whole thing can be ‘cubed’ and added ‘three’, to get a large prime.

Let us look at some examples:

1. 10000 — (10000 + 4)^ 3 + 3 = 1001200480067
2. 10001 — (10001 + 3)^ 3 + 3 = 1001200480067
3. 10011 — (10011 + 31)^ 3 + 3 = 1012652994091

Larger examples to print, because, when numbers are cubed, they give unexpectedly large results !
Got this one with difficulty —

100000 — (100000 + 12)^ 3 + 3 = 1000360043201731

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True
```

```
a=10000
b=2
while True:
    if primetest(((a+b)**3) + 3)==True:
        print(str(a), str(b), str(((a+b)**3) + 3))
        #print(str(a))
        #print(str(b))
        a+=1
        b=2
    else:
        b+=1
        if b**2>a:
            break
```

Conjecture 9 — The force or power of the double ton (part 1).

Conjecture — If x is a power of two, every odd number greater than x^2 can be written as x * a + b where a and b are prime (or 1)

This is a general observation for all powers of 2 (2, 4, 8, 16, 32, 64, 128, etc.). If 'x' is any of these, every odd number that is greater than x^2 can be written as x * a + b, where a and b are primes, or sometimes one.

So, for 2, every odd number greater than 4 can be written as 2 * prime + prime.
For 4, every odd number greater than 16 can be written as 4 * prime + prime
For 256, every odd number greater than 65536, can be written as 256 * prime + prime.

Some examples:-

2

1. 7 = 2 * 2 + 3
2. 21 = 2 * 7 + 7
3. 1133 = 2 * 563 + 7 (big)
4. 100135 = 2 * 50053 + 29 (very big)

64

1. 4097 = 64 * 61 + 193
2. 4251 = 64 * 61 + 347
3. 10313 = 64 * 73 + 5641 (big)
4. 19004225 = 64 * 296929 + 769 (very big)

To think that this works for every power of 2 is quite wonderful.

Here, its important to make a note. Take for example 4097. It can be written as 64 * a + b, because it is larger than 64 squared.
However, it can also be written as 2 * a + b, 4 * a + b, 8 * a + b, 16 * a + b, 32 * a + b, too, because, 4097 is greater than the squares of 2, 4, 8, 16 ad 32.

So an odd number can be written in multiple ways if there are many powers of two smaller than its square root.

Time to look at this code. The code is designed such that, it asks the user to give the power of 2. Based on the given number, it prints the outcomes for that particular power.

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
```

```
        y+=1
    if z!=1:
        return True

k=int(input('Enter k '))
#k=2
a=k**2 + 1
while True:
    b=3
    while True:
        c=a-b
        if c%k==0:
            if primetest(c/k)==True:
                print(str(a) + ' = ' + str(k) + ' * ' + str(c/k) + ' + ' + str(b))
                #print(a)
                break
            else:
                b+=2
            while primetest(b)==False:
                b+=2
        else:
            b+=2
            while primetest(b)==False:
                b+=2
    a+=2
    while a%k==0:
        a+=2
```

Conjecture 10 — The force or power of the double ton (part 2).

Conjecture — If x is a power of two, every odd number greater than x2 can be written as x * a - b where a and b are prime.**

Slightly different from conjecture 9. Instead of adding the second prime, we are subtracting it.

All other notes and properties remain, even when we do subtraction instead of addition.

Examples -

4

1. 17 = 4 * 5 - 3
2. 23 = 4 * 7 - 5
3. 1683 = 4 * 431 - 41 (big)
4. 153697 = 4 * 38459 - 139 (very big)

256

1. 65537 = 256 * 283 - 6911
2. 65665 = 256 * 257 - 127
3. 127063 = 256 * 587 - 23209 (big)
4. 12700205 = 256 * 49633 - 5843 (very big)
5. Code -

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

k=int(input('Enter k '))
#k=2
a=k**2 + 1
```

```
while True:
    b=3
    while True:
        c=a+b
        if c%k==0:
            if primetest(c/k)==True:
                print(str(a) + ' = ' + str(k) + ' * ' + str(c/k) + ' - ' + str(b))
                break
            else:
                b+=2
            while primetest(b)==False:
                b+=2
        else:
            b+=2
            while primetest(b)==False:
                b+=2
    a+=2
    while a%k==0:
        a+=2
```

Conjecture 11 — 2, 4, 6, is cricket's fate, primes are very great

Conjecture — For every prime number 'p' (other than 2), every even number greater than p^2 can be written as p * a + b, where a and b are also primes (as long as the even number is not a multiple of the prime number).

Now that we just discussed about powers of two, we look at something similar, but far more extraordinary, because, here we are dealing with primes, and only primes.

For every prime number x, every even number greater than its square can be written as x * p1 + p2, where p1 and p2 are also primes !.

So, for 3, every even number greater than 9 can be written as 3 * prime + prime.
For 5, every even number greater than 25 can be written as 5 * prime + prime
For 79, every even number greater than 6241, can be written as 79 * prime + prime.

Again, let us take separate examples:

3

1. 10 = 3 * 1 + 7
2. 20 = 3 * 5 + 5
3. 2956 = 3 * 983 + 7 (big)
4. 313714 = 3 * 104561 + 31 (very big)

107

1. 11450 = 107 * 101 + 643
2. 11614 = 107 * 103 + 593
3. 134658 = 107 * 1223 + 3797 (big)
4. 13453490 = 107 * 125693 + 4339 (very big)

Code — This code is such that, it'll start the computations from 3, and after every 100000 iterations, it'll switch automatically to the next prime number.

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True
```

```
#k=int(input('Enter k '))
k=3
itr=0
b=k**2 + 1
while True:
    while True:
        a=3
        while True:
            c=b-a
            if c%k==0:
                if primetest(c/k)==True:
                    print(str(b) + '=' + str(k) + '* ' + str(c/k) +' + ' + str(a))
                    itr+=1
                    if itr>100000:
                        k+=2
                        while primetest(k)==False:
                            k+=2
                        b=k**2 + 1
                        itr=0
                        break
                    #print(str(itr))
                    break
                else:
                    a+=2
                while primetest(a)==False:
                    a+=2
            else:
                a+=2
                while primetest(a)==False:
                    a+=2
        b+=2
        while b%k==0:
            b+=2
```

Conjecture 12 — Reach a milestone with a single or boundary

Conjecture — For every number a, greater than 17, there exists another number b, smaller than a, such that a+b is prime; a^2 + b^2 is also prime.

Another method to create two primes. For every number greater than 17, there exists a number smaller than it, such that, when the two numbers are added, we get a prime. But, when we add their squares, we still get a prime ! For every number, the number to be added can be different.

Some examples can clear the doubts.

1. 17 — 12; 17 + 12 is prime; 17^2 + 12^2 is prime
2. 18 — 5; 18 + 5 is prime; 18^2 + 5^2 is prime
3. 1732 — 27; 1732 + 27 is prime; 1732^2 + 27^2 is prime (big)
4. 170997 — 52; 170997 + 52 is prime; 170997^2 + 52^2 is prime (very big)

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=170992
```

```
b=3
while True:
    c=a**2
    d=b**2
    if primetest(c+d)==True and primetest(a+b)==True:
        print(str(a), str(b))
        #print(str(a))
        #print(str(d-c))
        a+=1
        b=3
    else:
        b+=1
        if d>c:
            break
```

Conjecture 13 — A boundary is just a square cut away

Conjecture — For every number greater than 117, the prime gap to the nearest prime, is always lesser than its square root.

This is probably the most controversial conjecture. It says that for any number, greater than 117, the nearest prime is less than the number's square root away. So a prime gap (distance between two primes), can never be larger than the square root of the number from where the gap starts.

Examples : - Here are some numbers and the gaps to the nearest prime.

1. 117, 10
2. 123, 4
3. 140, 9
4. 5671, 12 (big)
5. 56476782, 20 (very big)

This is the easiest conjecture to understand, yet I have a doubt on whether it can survive all the way unto infinity.

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=117
b=1
while True:
    if primetest(a+b)==True:
        print(str(a), str(b))
        #print(str(a))
        a+=1
        b=1
    else:
        b+=1
        if b**2>a:
            break
```

Conjecture 14 —Hole is a prime factor for a ball to get spoilt

Conjecture — Every number can be written as a^b + c, where a is a whole number, and b and c are prime. (or sometimes 1).

We again talk about powers. But, here, we are raising a number to the ‘prime’th power. So, take any number, and it can be written as something to the power prime plus a prime.

This works out for every single number, both odd and even. Sometimes, b and c can be one. This doesn't really make a big difference.

Some examples for this ‘power’ful conjecture. Works for numbers greater than 4.

1. 10 = 2 ^ 3 + 2
2. 28 = 3 ^ 2 + 19
3. 4007 = 2 ^ 2 + 4003 (big)
4. 56008 = 5 ^ 1 + 56003 (very big)

As we see, sometimes b can be one. But c always turns out to be a large prime number. Also, we can observe that the first number also tends to prime. This happens most of the time, but not always.

Eg - 4001 = 10 ^ 3 + 3001

Code -

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
```

```
            y+=1
    if z!=1:
        return True

x=6
a=2
b=1
c=2
while True:
    if x==a**b + c:
        #print(str(x))
        print(str(x) + ' = ' + str(a) + ' ^ ' + str(b) + ' + ' + str(c))
        #print(str(int((a+b)/2)))
        x+=1
        a=2
        b=1
        c=2
    else:
        c+=1
        if c>x:
            c=2
            b+=1
            while primetest(b)==False:
                b+=1
                while primetest(b)==False:
                    b+=1
                if a**b>x:
                    b=1
                    a+=1
        while primetest(c)==False:
            c+=1
```

Conjecture 15 — Odd square cuts

Conjecture — For every odd number a, greater than 10, there exists a prime number p, smaller than its half, such that, (a + p)^2 + 1 is also prime.

We saw something similar in Conjecture 7 which said *'For every number a, greater than 10000, there exists a number b, smaller than its square root such that, (a+b)^2 + 1 is prime.'*

However, here we are talking about odd numbers specifically, and those odd numbers greater than 10, not 10000. In conjecture 7, we said there exists a number b, smaller than its square root.
However, here, we specify, that there exists a 'prime' number smaller than its half.

Examples -
1. 11 — 3, (11 + 3)^2 + 1 is prime (197)
2. 27 — 13, (27 + 13)^2 + 1 is prime (1601)
3. 5469 — 5, (5469 + 5)^2 + 1 is prime (29964677)
4. 102441 — 19, (102441 + 19)^2 + 1 is prime (10498051601)

Once again, the first prime is very small compared to the large prime that is generated.

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
```

```
    if z!=1:
        return True

a=11
b=3
while True:
    if primetest(((a+b)**2) + 1)==True:
        print(str(a), str(b), str(((a+b)**2) + 1))
        #print(str(a))
        #print(str(b))
        a+=2
        b=3
    else:
        b+=2
        while primetest(b)==False:
            b+=2
        if b>a/2:
            break
```

Conjecture 16 — Odd amount of runs but reduce the doubles

Conjecture — For every odd number x, there exists a prime number P smaller than it, such that x * P - 2 is prime.

We continue about odd numbers. Again, we talk of the existence of a prime number, smaller than its half. We can multiply the odd number and the prime, and subtract 2 to get a large prime.

Examples:
1. 3 * 3 - 2 = 7
2. 5 * 3 - 2 = 13
3. 37 * 3 - 2 = 109
4. 3767 * 3 - 2 = 11299
5. 7706799 * 11 - 2 = 84774787

The formed primes are seriously gigantic. Straight forward conjecture. Its got a straight forward code.

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=3
b=3
while True:
    if primetest((a*b) - 2)==True:
        print(str(a), str(b), str((a*b) - 2))
        #print(str(a))
        a+=2
        b=3
    else:
        b+=2
        while primetest(b)==False:
            b+=2
        if b>a:
            break
```

Conjecture 17 — Hat trick and a Square cut, but no penalty

Conjecture — For every cube number (larger than 27), which is not divisible by 5, there exists a square number smaller than its half, such that it can be added to get a prime.

For the first time, we combine cubes and squares.

And after a bit of powers and multiplications, we take a break to come to simple addition.

We look at every cube number which is not a multiple of five. The conjecture states that there exists a square number smaller than it, such that when you add the cube and square, you get a prime.

Examples:- Lets look at some of the first few cubes that are not multiples of 5. We leave 2 cubed.

1. 27 + 4 = 31
2. 64 + 9 = 73
3. 216 + 25 = 241

Note that the first number is a cube, the second number is a square number smaller than the cube's half. When we add them, we get a nice little prime. :)
Let's go to the big ones.
456533 + 900 = 457433
22188041 + 2916 = 22190957

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
```

```
            return False
        else:
            y+=1
    if z!=1:
        return True

a=3
b=1
while True:
    if primetest(a**3 + b**2)==True:
        print(str(a**3), str(b**2), str(a**3 + b**2))
        a+=1
        while a%5==0:
            a+=1
        b=1
    else:
        b+=1
        if b**2>a**3:
            break
```

Conjecture 18 — 1000s for Steve and Mark Waugh

Conjecture — For every number a, greater than 1000, there exists a number b, smaller than its half, such that a*b is a number that is in between twin primes.

We focus on the numbers greater than 1000 for the next 6 conjectures.

Here, we say that, for every number ager 1000, there exists a number smaller than its half, which can be multiplied and added or subtracted to get a primes. In other words, generate twin primes.

For those of you who are not familiar with the term, twin primes are a pair of primes that differ by only 2. (eg: 3, 5). Using these conjecture, we can generate large pairs of twin primes.

So, lets look at the first 3 numbers after 1000.
1. 1000 * 3 = 3000
3001 is prime, 2999 is prime.

2. 1001 * 102 = 102102
102101 is prime, 102103 is prime.

3. 1002 * 5 = 5010
5009 is prime, 5011 is prime.

Some bigger ones —

7011 * 48 = 336528
336527 is prime, 336529 is prime.

703215 * 44 = 30941460
30941459 is prime, 30941461 is prime.
Note that the second number is smaller than the first number's half.

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True
```

```
a=1000
b=1
while True:
   if primetest(a*b + 1)==True and primetest(a*b - 1)==True:
      print(str(a) + ' * ' + str(b) + ' = ' + str(a*b) + ',  ' + str(a*b - 1),
str(a*b + 1))
      #print(a)
      #print(str(a))
      a+=1
      b=1
   else:
      b+=1
      if b>a/2:
         break
```

Conjecture 19 — The twin sixes

Conjecture — For every multiple of 6, greater than 1000, there exists a prime number smaller than its half which can be multiplied to get a number in between twin primes.

Similar to the previous one. We again incline to multiples of 6, because they literally work like magic. Here, the number to be multiplied is also prime. And the first number is a multiple of 6, greater than 1000. Again, we try to generate twin primes.

First few examples.

1. 1002 * 5 = 5010
5009 and 5011 are prime

2. 1008 * 31 = 31248
31247 and 31249 are prime

3. 1014 * 47 = 47658
47657 and 47659 are prime

Here, the second number is not just any number, but a prime number.

Bigger ones.
100320 * 29 = 2909280
2909279, 2909281 are prime

42400272 * 89 = 3773624208
3773624207, 3773624209 are prime

Code -

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=1002
b=3
while True:
    if primetest(a*b - 1)==True and primetest(a*b + 1):
        print(str(a) + ' * ' + str(b) + ' = ' + str(a*b) + ',  ' + str(a*b - 1),
str(a*b + 1))
        #print(a)
        a+=6
        if a==108:
```

```
        a+=6
    b=3
else:
    b+=2
    while primetest(b)==False:
        b+=2
    if b>a/2:
        break
```

Conjecture 20 — The consecutive fours (part 1)

Conjecture — For every odd number greater than 1000, there exists a multiple of four, smaller than its half, which can be added to get a prime and also added twice to get another prime.

We continue our journey with numbers greater than 1000. Here, we take a break from twin primes, and focus on generating two primes, not apart by 2, but by a multiple of 4.

This conjecture states that for every odd number greater than 1000, there exists a multiple of four, smaller than its half, which can be added to get a prime, and can be added twice to get another prime !!.
Examples —

1. 1001 + 48 = 1049, 1001 + 48 + 48 = 1097
2. 1003 + 60 = 1063, 1001 + 60 + 60 = 1123
3. 1005 + 4 = 1009, 1005 + 4 + 4 = 1013

Bigger ones —
34823 + 24 = 34847, 34823 + 24 +24 = 34871
94568075 + 276 = 94568351, 94568075 + 276 + 276 = 94568627

Note that the number added is always a multiple of four.

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=1001
b=4
while True:
    if primetest(a+b)==True and primetest(a+b+b)==True:
        print(str(a), str(b), str(a+b), str(a+b+b))
        a+=2
        b=4
    else:
        b+=4
        if b>a/2:
            break
```

Conjecture 21 — The consecutive fours (part 2)

Conjecture — For every odd number greater than 1000, there exists a multiple of four, smaller than its half, which can be subtracted to get a prime and also subtracted twice to get another prime.

Very similar to the previous one. Here we talk about subtracting a multiple of four once, then subtracting it again. Again, we are able to generate two primes.

Examples—
1. 1001 - 24 = 977, 1001 - 24 - 24 = 953
2. 1005 - 64 = 941, 1005 - 64 - 64 = 877
3. 1009 - 276 = 733, 1009 - 256 - 256 = 457

Bigger ones.
93001 - 180 = 92821, 93001 - 180 - 180 = 92641
78393057 - 44 = 78393013, 78393057 - 44 - 44 = 78392969

Note that the number subtracted is always a multiple of four.

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=1001
b=4
while True:
    if primetest(abs(a-b))==True and primetest(abs(a-b-b))==True:
        #print(str(a), str(b))
        print(str(a), str(b), str(a-b), str(a-b-b))
        #print(str(a))
```

```
        a+=4
        b=4
    else:
        b+=4
        if b>a/2:
            break
```

Conjecture 22 — Consecutive boundaries every 10 balls.

Conjecture — For every number a, greater than 1000, the nearest twin primes are away by a number less than a/10.

We wind up on twin primes. This is the second most unlikely and controversial conjecture.
It talks about the density of twin primes. The conjecture refers to the fact that, any number you pick, greater than 1000, the nearest twin prime pair is away by less than one tenth of the number picked.

So, if the number picked is 1000, the nearest twin prime pair is less than 100 away, as 100 is one tenth of 1000.
Some examples: —

1. For 1017, the nearest twin prime pair is away by 2 : 1019, 1021
2. For 1034, the nearest twin prime pair is away by 15 : 1049, 1051
3. For 34573, the nearest twin prime pair is away by 16 : 34589, 34591 (big)
4. For 99134590, the nearest twin prime pair is away by 181 : 99134771, 99134773 (very big !)

This conjecture would be very useful if it is proved to be true because, this can be used to prove that there are infinite number of twin prime pair; something which is still not definitely proven.

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=1000
b=1
while True:
    if primetest(a+b-1)==True and primetest(a+b+1):
        print(str(a), str(b-1), str(a+b-1), str(a+b+1))
        #print(a)
        a+=1
        b=1
    else:
        b+=1
        if b>a/10:
            break
```

Conjecture 23 — Run three less when hitting a square cut.

Conjecture — For every number a, greater than 1000, there exists a number b, smaller than its square root, such that, (a+b)^2 - 3 is prime.

We finish the conjectures related to numbers above 1000.
This one might be easy for those of you who have followed the previous ones closely. Its very straight forward and direct.

For every number greater than 1000, there exists a number, smaller than its square root, which can be added, the whole thing can be squared, and three can be subtracted from the product, to get a prime !

Examples —
1. (1000 + 4) ^ 2 - 3 = 1008013
2. (1023 + 25) ^ 2 - 3 = 1098301
3. (1059 + 17) ^ 2 - 3 = 1157773

Big ones —
(70367 + 69) ^ 2 - 3 = 4961230093
(970304 + 2) ^ 2 - 3 = 941493733633

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=1000
b=2
while True:
```

```
    if primetest(((a+b)**2) - 3)==True:
        print(str(a), str(b), str(((a+b)**2) - 3))
        #print(str(a))
        a#print(str(b))
        a+=1
        b=2
    else:
        b+=1
        if b**2>a:
            break
```

Conjecture 24 — Centuries without penalties.

Conjecture — For every number greater than 100, not a multiple of 5, there exists a multiple of 5, lesser than its half, which can be added to get a prime.

Another very easy conjecture, this time dealing with multiples of five.

For all numbers greater than 100, which aren't multiples of five, there exists a multiple of five, lesser than its half, which can be added to get a prime. Even as the numbers become large, we see that the multiple of five do not increase that drastically.

Examples —

1. 101 + 30 = 131
2. 102 + 5 = 107
3. 192 + 5 = 197
4. 10123 + 10 = 10133 (big)
5. 9820133 + 30 = 9820163 (very big)

Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=101
b=5
while True:
    if primetest(a+b)==True:
        print(str(a), str(b), str(a+b))
        #print(str(b))
        a+=1
        if a%5==0:
            a+=2
        b=5
    else:
        b+=5
        if b>a/2:
            break
```

Conjecture 25 — The Reverse Sweep

Note - This is the last conjecture and the one which I like the most. It is actually a kind of reverse of the Goldbach conjecture. Thus, the name — Reverse Sweep.

Conjecture — Every odd number above 13 can be written as the sum of two composite numbers, whose difference is prime (or rarely, 1).

Finally, my fellow Earthians; we reach the last conjecture. For me, this is the best :). Just a reminder:
The Goldbach conjecture goes like this — 'Every even number can be written as a sum of two primes'.

Well, I wanted to make a reverse conjecture for the Goldbach conjecture. That would probably be something like 'Every odd number is a sum of two composites'. But that was actually silly -_-.

So, I made it slightly more sensible.

This is the last conjecture 'Every odd number above 13 can be written as the sum of two composite numbers, whose difference is prime.' So, its not enough if the two numbers are composite.
Their difference has to be prime !.

Here are a few examples.
1. 13 = 9 + 4
2. 15 = 9 + 6
3. 35 = 27 + 8

Note that the two numbers that are added are composite and that their differences are prime.

Big ones

13239 = 13213 + 26
7713381 = 7713349 + 32
957713453 = 957713433 + 20

The Last Code —

```
def primetest(x):
    y=2
    z=0
    while y**2<=x:
        if x%y==0:
            z=1
            return False
        else:
            y+=1
    if z!=1:
        return True

a=13
b=a-1
while True:
    c=a-b
    if primetest(b)==False and primetest(c)==False and
primetest(b-c)==True:
        print(str(a) + ' = ' + str(b) + ' + ' + str(c))
        #print(str(a))
        a+=2
        b=a-1
    else:
        b=b-1
```

Epilogue —

So that's that. Thank you very much for your cooperation. Hope you had a good time. And, oh yeah.. a belated Happy New Year. The day I finished this book, was Jan 7. Coincidentally, 7 is a prime !
My thanks to my Parents, sister, uncles, aunts and Grandma for their constant support and encouragement; A big thanks to my math teachers, right from kindergarten:
Mrs. Leela
Mrs. Shobha
Miss Mohini
Mrs. Usha A
Mrs. Shakilakshi
Mrs. Anu Joshi
Mrs. Veena M
Mrs. Shailaja Bairy
Mrs. Uma Rao
Mrs. Divya
Mrs. Asha Mol

For guiding me for the last 12 years and making me love math.
My friend and teacher, Sri Vinay Nair, also has been a source of inspiration for me. I thank him too, along with Mr. Hariharan, Mr. Mandar, Prof. Venugopal Heroor and Mrs. Kusuma Kamath.

A big thanks to Dr. Arthur Benjamin, mathematician and magician from USA, for his encouragement and for being a tremendous source of inspiration. Thanks to my principal Mrs. Jessy Andrews and Vice Principal, Mrs. Surekha for their constant support. :)

I thank Mala ma'am once again for giving me that book, which was the beginning of all this. :) :) :)

And lastly, thank you dear reader, for spending your time on this book and hope it was a joyous (I know this is not the right word) read for you. Till next time, Ciao !!

- I finished this book at 8:29 PM, and once again, 829 happens to be a prime number, so thank you God for that small bit of magic.!