1. Computation will execute (2n-1) \* (1)

Where (2n-1) = times iterating through 1st loop on a cycle

And where (1) = times iterating through 2nd loop on a cycle

2) a) GCF(233, 144)

233 ÷ 144 = 1 Remainder 89

144 ÷ 89 = 1 Remainder 55

89 ÷ 55 = 1 Remainder 34

55 ÷ 34 = 1 Remainder 21

34 ÷ 21 = 1 Remainder 13

21 ÷ 13 = 1 Remainder 8

13 ÷ 8 = 1 Remainder 5

8 ÷ 5 = 1 Remainder 3

5 ÷ 3 = 1 Remainder 2

3 ÷ 2 = 1 Remainder 1

2 ÷ 1 = 2 Remainder 0

Thus, the GCD of 233 and 144 is 1

b) the numerical value of those would be 1 unless 0 is involved, as fibonacci numbers all only share a gcd of 1

c) f(n) = f(n-1) + f(n-2)

-----------------------

CORRECTION

GCD FOR BOTH IS = N-2

-----------------------

d)f(n) - f(n-1) - f(n-2)= 0

λ^n - λ^n-1 - λ^n-2= 0

λ ^2 - λ - 1= 0

Quadratic formula to get root of:

((1+/-SQRT(5))/2)

General solution is

c1((1+SQRT(5))/2)^n + c2((1+SQRT(5))/2)^n

f(n) =

n=0

f(0) = 2 =

2-=

f(1)=

1=+(2-)

1=+2-

1-2=-)

(1-2)/-) =

=1

= 2-1 = 1

f(n) = = ((1+SQRT(5))/2) + ((1+/-SQRT(5))/2)

3)

**a)** x= 3x

x=1

= 3

c=3c

c=⅓

=⅓

General solution

=⅓

Sub initial conditions, and solve

n=0

=⅔

Wolfram seems to compute that my answer is correct

**b)**x= 2x-1

x=0

= 2

-1

c=2c -1

-c=-1

c=1

General solution:

=1

substitute

=-1

Wolfram seems to compute that my answer is correct

**c)**x= 2x-1

x=1

= 2

-1

c=2c -1

-c=-1

c=1

General solution:

=1

substitute

=0

Wolfram seems to compute that my answer is correct

**d)**x= 5x-4n+1

=1

= 5

-4n+1

Guess : = n +

n + =5((n-1) + ) -4n + 1

n + =5n-5 + 5 -4n + 1

Pull out terms by their power

N:

n = 5n -4n

= 5n -4

-4 = -4

=1

1:

= -5+5 +1

= -5+5 +1

= -5+5 +1

= -4+5

=1

=n+

=n+

=n+

= + n+

Substitute

1 = + 0+

1 = +

=0

Wolfram seems to compute that my answer is correct

**e)**

x= x+2n+1

=1

= 1

+2n+1

Testing

Guess : = n

n=((n-1)) +2n + 1

n=n- +2n + 1

=-(/n)+2+1/n

-2 = (/n)+1/n

-2n=c1+1

-2n-1=c1

New stuff

= + n

=1= + 1\*0

=0

**f)**x= 2x+

7

=2

=c

=2x+

c=2(c) +

c=2(c) +3

c = 3

=

= +

Substitution

= +

7 = + 3

4 =

**g)**x= 2x+

7

=2

=c

=2x+

c=2(c) +

c2 = c2 + 2

BAD GUESS

=c+ d

c+ d = 2(c+ d) +

c+ d = 2c+ 2d +

c=

c=1

d = 2c+ 2d

D = 2(1)+ 2d

d=+2d

-d =

d=

c+ d = 2(c+ d) +

1+ = 2(1+ ) +

+ = 2(+ ) +

+ = + +

= 2\*+ +

= 2\*+

-( 2\*) =

0=0

=1\*+

=1\*+ +

=1\*+ +

7=1+ +

7=1-¼ +

28 = 4-1+4

25=4

=6.25

**4)**

Base Case S

t(3) > 0

t(2) > 0

t(1) > 0

Base case n=3

= ++1 = T(3) = T(2) + T(1) + T(0)

This is valid according to the question, so we are good to continue

Making the assumption that this stays true till k+1

Also making the assumption that the +1 is really

= ++ = T(k+1) = T(k) + T(k-1) + T(k-2)

++= T(k) + T(k-1) + T(k-2)

These are both effectively polynomials

Lambda^n ~=t(n) for equations

So we have:

(++)c = ()c >= T(k+1) = ++

Other Work: no solutions, but had a lot of trouble with this problem:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T(n) = T(n − 1) + T(n − 2) + T(n − 3)

base case S

T(3) > 0,

T(2) > 0,

T(1) > 0

T(3) = T(2) + T(1) + T(0)

summation from s=3 to n of T(n) = T(n − 1) + T(n − 2) + T(n − 3)

T(n+1) = T((T(n) = T(n − 1) + T(n − 2) + T(n − 3)) − 1) +

T((T(n) = T(n − 1) + T(n − 2) + T(n − 3)) − 2) +

T((T(n) = T(n − 1) + T(n − 2) + T(n − 3)) − 3)

λ03= λ02+λ01+1

λ0n= λ0(n-1)+λ0(n-2)+λ0(n-3)

λ03= λ0(3-1)+λ0(3-2)+λ0(3-3)

bigtheta(λ03) >= T(n) = T(n − 1) + T(n − 2) + T(n − 3)

(λ0(n-1)+λ0(n-2)+1)\*c=T(n − 1) + T(n − 2) + T(n − 3)

where n = 3, this is valid

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5)** t(h) = minimum number of the nodes in a height balanced tree of height h

Use induction

Base case note that t(1) >= t(0) >= 1

Thus, t(1) >= f and t(0) >= f

Assume t() >= for all < h

New avl tree with children - one child will be height h-1, other will be h-2. Minimum number of nodes in an AVL tree of height h can be bounded in terms of t(h-1) and t(h-3)

So, we can have

t(h)>=(h-1) + T(h-2)+1

This looks an awful lot like the fibonacci equation.

Further solving we approach a recurrence relation, where we calculate that the height will be equal to ~2x1.6^h

So this will be the minimum number of nodes in an AVL tree of height H

**t(h) > min # of nodes**

**t(h) > 2x1.6^h**

Reversing inequality

Will give us

2 × 1.6^h < t(h)

1.6^h < m(h)/2

log(1.6^h) < log(m(h)/2)

h log(1.6) < log(m(h)) - 1

0.678 h < log(m(h)) - 1

h < 1.475 log(m(h)) - 1.475

h < 2 log(m(h)) + 2

**So the max height for t entries would be appx.**

**t < 2 log(t) + 2**

**6)** Ogh would have to count up each individual notch on the stick, and while for small amounts this can be an alright system, its space usage on the stick is n/1 where n is the number of notches on the stick, whereas the arabic system uses what is effectively one notch per 10 numbers, or n/10 this by itself shows that Ogh’s system grows in storage size 10 times as fast as the arabic number system. Total counts would be a nightmare time-wise as well, because you will need to count each notch on the stick when creating the next part of the sequence instead of having a quicky referenced total value.

14 in arabic numerals is read a lot faster than IIIIIIIIIIIIII in notches