Problem 1

In [1]:
```
def reverseList(l, front, back):
    #base case
    if (back<=front):
        return
    temp=l[front]
    l[front]=l[back]
    l[back]=temp
    reverseList(l, front+1, back-1)

mylist = ['1','2','3','4','5','6','7','8','9','0','a','b','c','d','e']
print("Starting list 1: ",mylist)
reverseList(mylist, 0, len(mylist)-1)
print("Reversed list 1: ",mylist)

mylist2 =[]
print("Starting list 2: ",mylist2)
reverseList(mylist2, 0, len(mylist2)-1)
print("Reversed list 2: ",mylist2)

mylist3=[1,2,3]
print("Starting list 3: ",mylist3)
reverseList(mylist3, 0, len(mylist3)-1)
print("Reversed list 3: ",mylist3)

mylist4=['red','orange','yellow','green','blue','purple']
print("Starting list 3: ",mylist3)
reverseList(mylist3, 0, len(mylist3)-1)
print("Reversed list 3: ",mylist3)

mylist5=[5.8, 9.2, 2.2, 8.7]
print("Starting list 5: ",mylist5)
reverseList(mylist5, 0, len(mylist5)-1)
print("Reversed list 5: ",mylist5)
```

```
Starting list 1:  ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0', 'a', 'b', 'c', 'd',
'e']
Reversed list 1:  ['e', 'd', 'c', 'b', 'a', '0', '9', '8', '7', '6', '5', '4', '3', '2',
'1']
Starting list 2:  []
Reversed list 2:  []
Starting list 3:  [1, 2, 3]
Reversed list 3:  [3, 2, 1]
Starting list 3:  [3, 2, 1]
Reversed list 3:  [1, 2, 3]
Starting list 5:  [5.8, 9.2, 2.2, 8.7]
Reversed list 5:  [8.7, 2.2, 9.2, 5.8]
```

Problem 2

In [33]:
```
def srev(newlist,pos):
    if (len(newlist)==0):
        return []

    #Check if number at current index is a list
```

```
    if type(newlist[pos]) == list:
        newlist[pos]=srev(newlist[pos],0)

    #Check if next index exists. If it does, increase index and run again (Will recurse
    try:
        newlist[pos+1]
        newlist=srev(newlist,pos+1)
        return newlist

    except:
        newlist=newlist[::-1]
        return newlist

mylist=[1, [2 ,3], 4, [5, [6]]]
mylist2=[]
mylist3=[[1,2,[3,[8,[10,20,[23,88]]]]]]
mylist4=['a',['b','c',['d','e',['f','g','h']]],'i','j',['l',['m','n',['o','p',['q','r',
mylist5=['red','orange','yellow','green','blue','purple']

print("List1: ",mylist)
print("Reversed1: ",srev(mylist, 0))
print("List2: ",mylist2)
print("Reversed2: ",srev(mylist2, 0))
print("List3: ",mylist3)
print("Reversed3:",srev(mylist3,0))
print("List4: ",mylist4)
print("Reversed4:",srev(mylist4,0))
print("List5: ",mylist5)
print("Reversed5:",srev(mylist5,0))
```

```
List1:  [1, [2, 3], 4, [5, [6]]]
Reversed1:  [[[6], 5], 4, [3, 2], 1]
List2:  []
Reversed2:  []
List3:  [[1, 2, [3, [8, [10, 20, [23, 88]]]]]]
Reversed3: [[[[[[88, 23], 20, 10], 8], 3], 2, 1]]
List4:  ['a', ['b', 'c', ['d', 'e', ['f', 'g', 'h']]], 'i', 'j', ['l', ['m', 'n', ['o',
'p', ['q', 'r', 's']]]]]
Reversed4: [[[[['s', 'r', 'q'], 'p', 'o'], 'n', 'm'], 'l'], 'j', 'i', [[['h', 'g', 'f'],
'e', 'd'], 'c', 'b'], 'a']
List5:  ['red', 'orange', 'yellow', 'green', 'blue', 'purple']
Reversed5: ['purple', 'blue', 'green', 'yellow', 'orange', 'red']
```

Problem 3

Step 1 Define the Problem

This problem seems to be about the natural numbers so we let D = $\mathbb{N}$. We will define f, g, and P in the cells below.

In [34]:
```
def f3(n):
    if (n<=0):
        return 0
    return pow((n+2),3) - pow((n-1),3) + f3(n-1)
```

In [35]:
```
def P(n):
```

```
    return (f3(n)%9)==0

for i in range(10):
    print(i,P(i))
```

```
0 True
1 True
2 True
3 True
4 True
5 True
6 True
7 True
8 True
9 True
```

Step 2 Check the Stopping Value + 2 More

We already did that when we ran 10 values of P for n in range(10) which includes the Stopping Value 0.

Step 3 Inheritance

We are allowed to assume P(n-1) is True and we want to prove that P(n)%9=0 is True. In other words, if we assume f(n-1)%9=0, then we want to prove that f(n)%9=0 is True. We can show this by starting with f(n-1) and turning it into f(n) with just terms that are all divisible by 9. Here are the steps and the reasons.

Statement	Reason
f(n-1)=$(n-1)^3$+n^3+$(n+1)^3$	Defintion of f(n-1)
f(n-1)=n^3-3n^2+3n-1+n^3+$(n+1)^3$	Distribute $(n-1)^3$
f(n)=n^3+$(n+1)^3$+$(n+2)^3$	Definition of f(n)
f(n)=n^3+$(n+1)^3$+n^3+6n^3+12n+8	Distribute $(n+2)^3$
f(n)=n^3+$(n+1)^3$+$(n-1)^3$+9n^2+9n+9	Definition of f(n-1)
f(n)=f(n-1)+9(n^2+n+1)	Factor out a 9

Step 4 Conclude the Proof

Because we verified steps 1 through 3, by Induction it follows that for any natural number n, f(n)%9=0

Problem 4

Step 1 Define the Problem

This problem seems to be about the natural numbers so we let D = $\mathbb{N}$. We will define f, g, and P in the cells below.

In [36]:
```
def f(n):
    if (n<=0):
        return 0
    return (n*n) / ((2*n-1)*(2*n+1)) + f(n-1)

for i in range(10):
    print(i,f(i))
```

```
0 0
1 0.3333333333333333
2 0.6
3 0.8571428571428571
4 1.1111111111111112
5 1.3636363636363638
6 1.6153846153846154
7 1.8666666666666667
8 2.1176470588235294
9 2.3684210526315788
```

In [37]:
```
def g(n):
    return (n*(n+1))/(2*(2*n+1))

for i in range(10):
    print(i,g(i))
```

```
0 0.0
1 0.3333333333333333
2 0.6
3 0.8571428571428571
4 1.1111111111111112
5 1.3636363636363635
6 1.6153846153846154
7 1.8666666666666667
8 2.1176470588235294
9 2.3684210526315788
```

In [38]:
```
def P(n):
    return round(f(n),5)==round(g(n),5)

for i in range(10):
    print(i,P(i))
```

```
0 True
1 True
2 True
3 True
4 True
5 True
6 True
7 True
8 True
9 True
```

Step 2 Check the Stopping Value + 2 More

We already did that when we ran 10 values of P for n in range(10) which includes the Stopping Value 0.

Step 3 Inheritance

We are allowed to assume P(n-1) is True and we want to prove that P(n) is True. In other words, if we assume f(n-1) = g(n-1) then we want to prove that f(n) = g(n). Here are the steps and the reasons.

Statement	Reason
f(n) = (n^2)/((2n-1)(2n+1)) + f(n-1)	Definition of f
f(n) = (n^2)/((2n-1)(2n+1)) + g(n-1)	P(n-1) is true
f(n) = (n^2)/((2n-1)(2n+1)) + (n-1)(n)/(n(2n-2+1))	Defintion of g(n-1)
f(n) = n(n/((2n-1)(2n+1)) + (n-1)/(2(2n-1))	factor out n
f(n) = n((2n+(2n+1)(n-1))/(2(n-1)(2n+1))	create a common denominator
f(n) = (n/(2(2n+1))*(2n+2n^2+n-2n-1)/(2n-1))	factor out common factors
f(n) = (n/(2(2n+1))*(2n^2+n-1)/(2n-1))	combine like terms
f(n) = (n/(2(2n+1))*((n+1)(2n-1)/(2n-1))	factor numerator
f(n) = (n/(2(2n+1))*(n+1)	cancel like factors
f(n) = g(n)	Definition of g

So P(n) is True. Every step except for the inductive hypothesis is either a definition or Algebra.

Step 4 Conclude the Proof

Because we verified steps 1 through 3, by Induction it follows that for any natural number n, f(n) = g(n).

Problem 5

Problem 5a

In [24]:
```
def rp(alph,s):
    alen=len(alph) #length of the alphabet
    slen=len(s) #length of the string
    listsize=pow(alen,slen)
    order=['']*listsize #makes an empty list of desired length


    b=pow(alen,slen-1) #num times to consecutively print letter
    c=0; #pos of thing in list
    i=slen; #num of cols

    while i>0: # for every column
        while (c<listsize): #do pos x for every element in list 0-len
```

```
            for a in range(0,alen): #go through every letter in the alphabet
                btemp=b
                while (btemp>=1): #consecutively print letter
                    order[c]+=alph[a]
                    c+=1
                    btemp-=1
        i-=1
        c=0
        b//=alen #decrease the number of times consecutively printing letter by desired

    #print(order)

    return order.index(s)

print("('abc'", ",'bc')", rp('abc','bc'))
print("('cab'", ",'bc')", rp('cab','bc'))
print("('abc'", ",'babcac')", rp('abc','babcac'))
print("('theory'", ",'oo')", rp('theory','oo'))
print("('wxyz'", ",'wxy')", rp('wxyz','wxy'))
```

```
('abc' ,'bc') 5
('cab' ,'bc') 6
('abc' ,'babcac') 290
('theory' ,'oo') 21
('wxyz' ,'wxy') 6
```

Problem 5b

In [62]:

```
def rp(alph,s):
    alen=len(alph)
    slen=len(s)
    listsize=pow(alen,slen)
    order=['']*listsize


    b=pow(alen,slen-1) #num times to consecutively pring letter
    c=0; #pos of thing in list
    i=slen;

    while i>0:
        while (c<listsize):
            for a in range(0,alen):
                btemp=b
                while (btemp>=1):
                    order[c]+=alph[a]
                    c+=1
                    btemp-=1
        i-=1
        c=0
        b//=alen

    return order #returns list instead of position

def s2n(alph,s):
    l=[]
    slen=len(s)
    i=1
    while (i<=slen):
        l+=rp(alph,s[0:i])
```

```
        i+=1

    return l.index(s)+1

print("('abc'", ",'babcac')", s2n('abc','babcac'))
print("('abc'", ",'bc')", s2n('abc','bc'))
print("('cab'", ",'bc')", s2n('cab','bc'))
print("('theory'", ",'oo')", s2n('theory','oo'))
print("('wxyz'", ",'wxy')", s2n('wxyz','wxy'))
```

```
('abc' ,'babcac') 654
('abc' ,'bc') 9
('cab' ,'bc') 10
('theory' ,'oo') 28
('wxyz' ,'wxy') 27
```

Problem 5c

In [4]:

```
def n2Lnp(alph,n):
    x=len(alph) #multiple of alpha lenth
    c=1 #first return value
    s=0 #sum
    y=x #product of powers

    r=n

    while (s<n):
        s+=y
        c+=1
        if s<n:
            y*=x

    c-=1

    i=c-1
    while (i>=1):
        r-=(pow(x,i))
        i-=1

    r-=1
    print("(",c,",",r,")")

print("alphabet: abc, number: 654")
n2Lnp('abc',654)
print("alphabet: abc, number: 9")
n2Lnp('abc',9)
print("alphabet: cab, number: 10")
n2Lnp('cab',10)
print("alphabet: theory, number: 28")
n2Lnp('theory',28)
print("alphabet: wxyz, number: 27")
n2Lnp('wxyz',27)
```

```
alphabet: abc, number: 654
( 6 , 290 )
alphabet: abc, number: 9
( 2 , 5 )
alphabet: cab, number: 10
( 2 , 6 )
alphabet: theory, number: 28
```

```
( 2 , 21 )
alphabet: wxyz, number: 27
( 3 , 6 )
```

Problem 5d

In [6]:

```
def rp(alph,s):
    alen=len(alph)
    slen=len(s)
    listsize=pow(alen,slen)
    order=['']*listsize


    b=pow(alen,slen-1) #num times to consecutively pring letter
    c=0; #pos of thing in list
    i=slen;

    while i>0:
        while (c<listsize):
            for a in range(0,alen):
                btemp=b
                while (btemp>=1):
                    order[c]+=alph[a]
                    c+=1
                    btemp-=1
        i-=1
        c=0
        b//=alen

    return order

def n2s(alph, n):
    multiple = len(alph)
    prod = 1
    s = 0
    string = ''
    L = []
    while (s<n):
        string+='a'
        prod*=multiple
        s+=prod
        L+=rp(alph,string)
    return L[n-1]

print("('abc',654)",n2s('abc',654))
print("('abc',9)",n2s('abc',9))
print("('cab',10)",n2s('cab',10))
print("('theory',28)",n2s('theory',28))
print("('abc',9)",n2s('abc',9))
print("('wxyz',27)",n2s('wxyz',27))
```

```
('abc',654) babcac
('abc',9) bc
('cab',10) bc
('theory',28) oo
('abc',9) bc
('wxyz',27) wxy
```

Problem 5e

In [7]:

```
import random #used for random number generation
import string

#used in n2s and s2n
def rp(alph,s):
    alen=len(alph)
    slen=len(s)
    listsize=pow(alen,slen)
    order=['']*listsize


    b=pow(alen,slen-1) #num times to consecutively pring letter
    c=0; #pos of thing in list
    i=slen;

    while i>0:
        while (c<listsize):
            for a in range(0,alen):
                btemp=b
                while (btemp>=1):
                    order[c]+=alph[a]
                    c+=1
                    btemp-=1
        i-=1
        c=0
        b//=alen

    return order

def n2s(alph, n):
    multiple = len(alph)
    prod = 1
    s = 0
    string = ''
    L = []
    while (s<n):
        string+='a'
        prod*=multiple
        s+=prod
        L+=rp(alph,string)
    return L[n-1]

def s2n(alph,s):
    l=[]
    slen=len(s)
    i=1
    while (i<=slen):
        l+=rp(alph,s[0:i])
        i+=1

    return l.index(s)+1

lets = string.ascii_letters
alph1=random.choice(lets)+random.choice(lets)+random.choice(lets)
alph2=random.choice(lets)+random.choice(lets)+random.choice(lets)
alph3=random.choice(lets)+random.choice(lets)+random.choice(lets)
s1=''
s2=''
s3=''
```

```
s4=''
s5=''
for i in range(12):
    s1+=alph1[random.randint(0,2)]
    s2+=alph1[random.randint(0,2)]
    s3+=alph1[random.randint(0,2)]
for i in range(12):
    s4+=alph2[random.randint(0,2)]
for i in range(12):
    s5+=alph3[random.randint(0,2)]

num1=s2n(alph1,s1)
print("Alphabet:",alph1,"string:",s1)
print("Position:",num1,"Result:",n2s(alph1,num1))
print()
num2=s2n(alph1,s2)
print("Alphabet:",alph1,"string:",s2)
print("Position:",num2,"Result:",n2s(alph1,num2))
print()
num3=s2n(alph1,s3)
print("Alphabet:",alph1,"string:",s3)
print("Position:",num3,"Result:",n2s(alph1,num3))
print()
num4=s2n(alph2,s4)
print("Alphabet:",alph2,"string:",s4)
print("Position:",num4,"Result:",n2s(alph2,num4))
print()
num5=s2n(alph3,s5)
print("Alphabet:",alph3,"string:",s5)
print("Position:",num5,"Result:",n2s(alph3,num5))
```

```
Alphabet: xvx string: vxvvxvxxvxxx
Position: 469867 Result: vxvvxvxxvxxx

Alphabet: xvx string: xxvxxxxxvxxx
Position: 285430 Result: xxvxxxxxvxxx

Alphabet: xvx string: xxxvxxxxxvxv
Position: 272291 Result: xxxvxxxxxvxv

Alphabet: RGB string: GBGRGRGBRBBG
Position: 583265 Result: GBGRGRGBRBBG

Alphabet: TBV string: BBBTTTVBVVVV
Position: 522246 Result: BBBTTTVBVVVV
```

Strings of length 12 was the highest I could get to run without the program taking forever. When trying to make the range of the loops 20, it seemed as if the program would never end. So, I need to optimize this code, and there is probably a better way to do the np method.

Problem 6

In [10]:

```
def fib(n):
    if n < 2:
        return 1
    return fib(n-1) + fib(n-2)
```

In [11]:

```
def fibmax(n):
    if n < 2:
        return [1,1]
    L = fibmax(n-1)
    if L[-1] + L[-2] > n:
        return L
    return L+ [L[-1]+L[-2]]
```

Problem 6a

Step 1 Define the Problem

This problem conserns natural numbers from j to p starting at 0, so we let D = $\mathbb{N}$. We will define fib, fibmax, and P in the cells below.

In [14]:
```
def Pa(n):
    L=fibmax(n)
    for j in range (len(L)):
        print(fibmax(n)[j],fib(j))
        print(fibmax(n)[j]==fib(j))

Pa(25)
```

```
1 1
True
1 1
True
2 2
True
3 3
True
5 5
True
8 8
True
13 13
True
21 21
True
```

Step 2 Check the Stopping Value + 2 More

We already did that when we ran 25 values of p for j in range(25) which includes the stopping value 0

Step 3 Check that Recursion stay in D

The domain is all natural numbers. So, given a positive integer p, note that fib(p) calls fib(p-1) and fib(p-2). If p>=2, then fib(p) will eventuall stop because the base case is p<2. Due to recursively calling p-1 and p-2, p must keep getting smaller and smaller until p<2 and returns 1. Also, if you are given the positive interger 1, the fib(1), fib(0), and fib of (-1) will all return 1. So if you are given any positive integer, wether that be 1 or something >=2, fib(p) forces itself to decrease and stop when

p<2. Due to this recursion and stopping value of p<2, all values in the domain of p are positive integers. Next, we notice that fibmax(p) calls fibmax(p-1) when creating L and also has a base case of p<2. If you are given a postive integer p>=2 this recursive decrement of 1 will eventually call fibmax(1) which will return [1,1] and stop recursion. If you are given the last possible positive integer, p=1, then p(1) will automatically be called and return [1,1]. Due to this recursive decreasing of p and base case of p<2, all values in the domain of p are integers or natural numbers.

Step 4 Prove that Recursion Halts

We will use the counting strategy. We need to assign a non-negative integer to elements of N which is the set of all natural numbers. The counter decreases for fib(p) because it calls fib(p-1) and fib(p-2). The counter would also decrease for fibmax(p) because L=fibmax(p-1) calls fibmax(p-1). Since both these counters decrease, recursion must hault.

Step 5 Prove that P is inherited Recusively

Let p be a positive integer and L=fibmax(p). I have to prove that P(n) is true if P(n-1) is true. It is assumed that P(n-1) is true. P(n-1) produces a maxium number of calls until n=1 since 1 or n<2 is the base case and both methods recursively call n-1. P(n) would be true because both fibmax(n) and fib(n) recursively call fibmax(n-1) and fib(n-1) which is known to be true. Since, P(n) is just calling these two functions which we already know to be true or P(n-1), then we can assume P(n) is also true.

Step 6 Conclusion

Since we have shown that steps 1-5 are satisfied, we have proven that for any n$\in N$, recursion will eventually stop due to both functions calling n-1 and that L[j]==fib(j) for j in range len(L).

Problem 6b

Step 1 Define the Problem

This problem conserns natural numbers from n to p starting at 1, so we let D = $\mathbb{N}$. We will define P in the cell below. Fib and fibmax are defined at the beginning of problem 6.

In [28]:
```
def Pb(p):
    L=fibmax(p)
    tf=True
    for n in range(len(L)):
        if (L[n]>0 and fib(n)<=p):
            tf=True
        else:
            return False
    return tf

for p in range(0,15):
    print(p,Pb(p))
```

```
0 False
1 True
2 True
3 True
4 True
5 True
6 True
7 True
8 True
9 True
10 True
11 True
12 True
13 True
14 True
```

Step 2 Check the Stopping Value + 2 More

We already did that when we ran 15 values of p for n in range(15) which includes the stopping value 0

Step 3 Check that Recursion stay in D

The domain is all natural numbers.Given a non-negative integer, both fib(n) and fibmax(n) will trigger recursion as long as n>=2. So, fib(n-1), fib(n-2), and fibmax(n-1) will be called. These calls are in the domain of natural numbers because when n>=2, n-1>=1, and n-2>=0. Further, 2,1, and 0 are all natural numbers.

Step 4 Prove that Recursion Halts

Recursion will halt for both fib(n) and fibmax(n) when they reach their base case of n<2. Fib(n) will return 1 and fibmax(n) will return [1,1] when these cases are reached without calling another function.

Step 5 Prove that P is inherited Recusively

We assume P(n-1) is true when proving P(n) is true. We can prove that P(n)<=p by using the definiton of fibonacci f(n)=f(n-1)+f(n-2). Since we know all fibonacci below n is true, we can replace them with L. Further L[n] has to be at least 1 according to the base case.

Step 6 Conclusion

Since we have shown that steps 1-5 are satisfied, we conlude that L[n]>0 and fib(n)<=p for our domain of all natural numbers.

Problem 6c

Step 1 Define the Problem

This problem conserns natural numbers from n to p starting at 1, so we let D = $\mathbb{N}$. We will define P in the cell below.Fibmax is defined at the beginning of problem 6.

In [40]:
```
import math #log and sqrt
def Pc(p):
    ratio=(1+math.sqrt(5))/2
    lhs=len(fibmax(p))
    rhs=math.log(p,ratio)+2
    if lhs<=rhs:
        return True
    return False

for p in range(1,11):
    print(p,Pc(p))
```

```
1 True
2 True
3 True
4 True
5 True
6 True
7 True
8 True
9 True
10 True
```

Step 2 Check the Stopping Value + 2 More

We already did that when we ran 15 values of p for n in range(15) which includes the stopping value 1

Step 3 Check that Recursion stay in D

The domain is all natural numbers.Given a non-negative integer, fibmax(n) will trigger recursion as long as n>=2. So, fibmax(n-1) will be called. These calls are in the domain of natural numbers because when n>=2 and n-1>=1 both 1 and 2 are in the domain of natural numbers

Step 4 Prove that Recursion Halts

Recursion will halt for fibmax(n) when it reachs the base case of n<2. Fibmax(n) will return [1,1] when this case is reached without calling another function.

Step 5 Prove that P is inherited Recusively

We assume P(n-1) is true when proving P(n) is true. We can prove that P(n)<=p by using the definiton of fibonacci f(n)=f(n-1)+f(n-2). Since we know all fibonacci below n is true, we can replace them with L. Further, the length of L can only be as long as the number of fibonacci numbers below n, so it seems reasonable to assume that the length will also be less than or equal to the log base of the golden ratio of p plus 2.

Step 6 Conclusion

Since we have shown that steps 1-5 are satisfied, we conlude that the length of L will be less than or equal to the log base of the golden ratio of p plus 2 for our domain of all natural numbers.

Problem 7

Statement	Reason	
Let Q={$q	a+b=a$}	Definition
0 is in Q because if a=0 & b=0	Defintion of addition	
0+0=0		
Suppose q∈Q	4th Postulate	
(q+b)'=q+b'	Definiton of addition	
q+b'=q'		
b'=0'		
So, q∈Q iff b=0		
So, a+b=0 iff b=0		

Problem 8

Statement	Reason	
Let Q={$(a,b)	a+b$=0}	Defintion
(0,0) is in Q	Definition of Addition	
Assume a∈Q so a+b=0		
is a' in Q		
for b=0, 0+q'=q'	Lemma 1	
q'≠0 & q'≠Q	2nd Postulate	
Q={(0,0)}	4th Postulate	
a+b=0 iff a=0 & b=0	b/c a=0 & Q{(0,0)}	