Problem 1

In [38]:
```
import math as m
List1= [1,5,3,4,5,6,7,8,9,10,11,12]
def evenElementsrem(L):
    listcured=[]
    for elements in range (0,len(L)):
        if elements %2 == 0:#whenever the index is a multiple of 2 :0,2,2n
            listcured.append(L[elements])#the number is appended to the li
    return listcured
print(evenElementsrem(List1))
```

```
[1, 3, 5, 7, 9, 11]
```

Problem 2

In [39]:
```
def hailstoneseq(list):
    i=0
    j=0
    while i!=1:
        if list[len(list)-1]%2==0 :#checks if the last element in the list
            list.append(list[len(list)-1]/2)
            i=list[len(list)-1]
        else:#in case it is odd
            list.append(3*list[len(list)-1]+1)
            i=list[len(list)-1]
        j+=1#the counter of the attempts it takes for the number to reach

    return j

def collatzconj():
    tries_convg=[]
    for i in range (1,100):
        n=[i]
        k=hailstoneseq(n)
        tries_convg.append(k)#counts the number of tries for numbers 1 t0
    return tries_convg
print(collatzconj())
```

```
[3, 1, 7, 2, 5, 8, 16, 3, 19, 6, 14, 9, 9, 17, 17, 4, 12, 20, 20, 7, 7, 1
5, 15, 10, 23, 10, 111, 18, 18, 18, 106, 5, 26, 13, 13, 21, 21, 21, 34,
8, 109, 8, 29, 16, 16, 16, 104, 11, 24, 24, 24, 11, 11, 112, 112, 19, 32,
19, 32, 19, 19, 107, 107, 6, 27, 27, 27, 14, 14, 14, 102, 22, 115, 22, 1
4, 22, 22, 35, 35, 9, 22, 110, 110, 9, 9, 30, 30, 17, 30, 17, 92, 17, 17,
105, 105, 12, 118, 25, 25]
```

Problem 3

In [40]:
```
list=[0,1,2]
for l in enumerate(list,10):
    print(l)
```

```
(10, 0)
(11, 1)
(12, 2)
```

Problem 4

In [41]:
```
def fact(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return fact
print(fact(2))
```

```
2
```

Problem 5

In [42]:
```
def tetration(n,x):
    if n==0:
        return 1
    else:
        return x**tetration(n-1,x)
len(str(tetration(5,2)))
```

Out[42]:
```
19729
```

Problem 6

In [43]:
```
from math import sqrt
def f(x,nmax=53):
    for i in range(nmax):
        x = sqrt(x)
    for i in range(nmax):
        x = x**2
    return x
for xin in (5., 0.5):
    xout = f(xin)
    print(xin, xout)
```

```
5.0 1.0
0.5 0.3678794384340895
```

Square rooting a number greater than 1 several times brings it very close to 1 such that the number is essentially rounded off to 1 as it is 1 up to several decimal places.Squaring one,even doing so a hundred times we start getting 1 as $(1^2)^\infty = 1$.

Similarly, for a number less than one, square rooting it brings it very close to zero such that the

number is rounded off to zero and squaring zero also gives zero.

Problem 7

$a)$

$$x_+x_- = \frac{(-b+\sqrt{b^2-4ac})(-b-\sqrt{b^2-4ac})}{4a^2}$$

$$x_+x_- = \frac{b^2-b^2+4ac}{4a^2}$$

$$x_+x_- = \frac{c}{a} shown$$

In [44]:
```
def bad_root_qdeqn(a,b,c):
    x_pl=(-b+sqrt((b**2)-4*a*c))/2*a
    x_mi=(-b-sqrt((b**2)-4*a*c))/2*a
    list=[x_pl,x_mi]
    return list
def good_root_qdeqn(a,b,c):
    x_mi=(-b-sqrt(b**2-4*a*c))/2*a
    x_pl=c/(x_mi*a)
    list=[x_pl,x_mi]
    return list
print(bad_root_qdeqn(1,108,1),good_root_qdeqn(1,108,1))
```

```
[-0.009260053227649223, -107.99073994677235] [-0.0092600532276461, -107.9
9073994677235]
```

We see that the answers are correct to 13^{th} decimal place for x_+ and the start deviating from there; whereas, the x_- is correct all the way.

Problem 8

part a

In [45]:
```
x_int =1234567891234567
y_int=1234567891234566
(x_int**2)-(y_int**2)
```

Out[45]: 2469135782469133

part b

In [46]:
```
x_fl =1234567891234567.0
y_fl=1234567891234566.0
dirdiff=(x_fl**2)-(y_fl**2)
diff_sq=(x_fl-y_fl)*(x_fl+y_fl)
print(dirdiff," ",diff_sq)
```

```
2533274790395904.0   2469135782469133.0
```

Problem 9

Part a&b

In [96]:
```
def riemannapprox():
    prevsum=1
    summ=0
    i=1
    term=0
    while (m.isclose(prevsum,summ)) is False:
        prevsum=summ
        term=(1/i**2 )
        summ+=term
        i=i+1
    nmaxd = i
    list=[summ,prevsum,nmaxd]
    return list
print(riemannapprox())
```

```
[1.6448935112363525, 1.6448935095915282, 24658]
```

$$lim_{k \to 24658} \frac{1}{k^2} \approx 0$$

Part c

In [95]:
```
nmaxd=riemannapprox()[2]
nmaxr=1024*nmaxd
summ=0
while(nmaxr!=0):
    summ+=1/nmaxr**2
    nmaxr-=1
print(summ)
```

```
1.6449340272439406
```

part d

In [ ]:
```

```