Exam 2

This exam is confidential and is not to be shared with anyone not in ChEn 263 Fall 2021

Problem 1 (18 points)

If I take two numbers x_1 and x_2 then the *arithmetic* average is

$$\bar{x}_a = \frac{1}{2}(x_1 + x_2)$$

the *geometric* average is

$$\bar{x}_g = (x_1\, x_2)^{\frac{1}{2}}$$

and the *harmonic* average is

$$\bar{x}_h = 2\big(x_1^{-1} + x_2^{-1}\big)^{-1}$$

These three averages each has uses in physics and engineering.

Make a function that computes and prints all three types of averages of any two non-negative numbers. There should be a check to warn the user and gracefully stop computation if one or both input numbers is *negative*. In the case of the harmonic average you additionally need to make the code correctly compute the result of zero when one or both of the input numbers is zero. Test your function by calling it in turn for the following three sets of numbers.

- -5 and 14
- 0 and 9
- 3 and 6

The function should output with detailed print statements so the user can tell just by reading the output what two numbers are being averaged, what kind of average is being computed, and the average value. Also make sure your code itself is reasonably documented.

In [1]:

```
def triple_avg(x1, x2):
    # Catch negative arguments
    if(x1 < 0 or x2 < 0):
        print(x1, "and", x2, "are not valid arguments. The arguments cannot be negative
        return

    # Calculate and print the arithmetic average
    arithmetic_avg = (x1 + x2)/2
    print("The arithmetic average of ",x1,"and",x2,"is",arithmetic_avg)

    # Calculate and print the geometric average
    geometric_avg = (x1*x2)**0.5
    print("The geometric average of ",x1,"and",x2,"is",geometric_avg)

    # Calculate and print the harmonic average, setting it first to zero
    harmonic_avg = 0
    # then only recalculating it's value if x1 and x2 are both non-zero
    if(x1 != 0 and x2 != 0):
        harmonic_avg = 2/(1/x1 + 1/x2)
    print("The harmonic average of ",x1,"and",x2,"is",harmonic_avg)

    return

triple_avg(-5, 14)
triple_avg(0, 9)
triple_avg(3, 6)
```

```
-5 and 14 are not valid arguments. The arguments cannot be negative numbers.
The arithmetic average of  0 and 9 is 4.5
The geometric average of  0 and 9 is 0.0
The harmonic average of  0 and 9 is 0
The arithmetic average of  3 and 6 is 4.5
The geometric average of  3 and 6 is 4.242640687119285
The harmonic average of  3 and 6 is 4.0
```

Problem 2 (18 points)

In Homework 10 Problem 1 you made a custom function that computes the Gaussian error function $\operatorname{erf}(x)$ using a Taylor series centered on $x = 0$. For a finite number of terms the resulting function starts to develop accuracy problems if $|x|$ is large. An alternative way to construct the needed function for large $|x|$ values and a relatively small number of terms is with an *asymptotic expansion*, which is kind of like a Taylor series except centered on $|x| \to \infty$. The formula for the asymptotic expansion in this case is

$$\operatorname{erf}(x) = 1 - \frac{\exp(-x^2)}{\sqrt{\pi}\,x} \sum_{n=0}^{\infty} \left[\prod_{k=1}^{n} \frac{1-2k}{2x^2} \right]$$

Implement this function in well-documented Python code and use 14 terms in the infinite series (i.e. `n_max = 13`). Then call your function and print out the value of $1 - \operatorname{erf}(3.5)$

You might want to make sure your function is working properly by comparing results to the built-in `math.erf` function (available once you `import math` in your code).

In [2]:

```
import math
from numpy import exp, pi

def my_erf(x):
    # set constant n_max
    n_max = 13

    # calculate the sum of products
    sum = 0
    for n in range(n_max + 1): #ranging from 0 to n_max (inclusive)
        mult = 1
        for k in range(1, n + 1): #ranging from 0 to n (inclusive)
            mult *= (1 - 2*k)/(2*x**2)
        sum += mult
    # plug the sum into the given equation and return the answer
    answer = 1 - exp(-1 * x**2)/(pi**0.5*x)*sum
    return answer

print("1 - my_erf(3.5) =", 1 - my_erf(3.5))
print("1 - math.erf(3.5) =", 1 - math.erf(3.5))
```

```
1 - my_erf(3.5) = 7.430956330267691e-07
1 - math.erf(3.5) = 7.4309837239106e-07
```

Problem 3 (9 points)

Use Latex within Markdown to reproduce precisely the series formula for $\mathrm{erf}(x)$ displayed in Problem 2. Every character and symbol should look exactly the same: font, style, position, and relative size. The equation as a whole should be similarly positioned relative to surrounding text.

$$\mathrm{erf}(x) = 1 - \frac{\exp\left(-x^2\right)}{\sqrt{\pi}x} \sum_{n=0}^{\infty} \left[\prod_{k=1}^{n} \frac{1-2k}{2x^2} \right]$$

Note: You must submit your exam as a pdf file, but first make sure it is properly displaying the Markdown and results from your Python code. To this end, do the following

1. Run $\longrightarrow$ Restart Kernel and Run All Cells
2. Covert the ipynb file to html (File $\longrightarrow$ Export Notebook As $\longrightarrow$ HTML)
3. Open the html file in your browser and *print to pdf*. Make sure file name has been changed to your personal name.

In [ ]:

```

```