In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colors
import math
import seaborn as sns

print("Part a: \n")
print("x^20")
print("-210 x^19")
print("20615 x^18")
print("-1256850 x^17")
print("53327946 x^16")
print("-1672280820 x^15")
print("40171771630 x^14 ")
print("-756111184500 x^13")
print("11310276995381 x^12")
print("-135585182899530 x^11")
print("1307535010540395 x^10")
print("-10142299865511450 x^9")
print("63030812099294896 x^8")
print("-311333643161390640 x^7")
print("1206647803780373360 x^6 ")
print("-3599979517947607200 x^5")
print("8037811822645051776 x^4")
print("-12870931245150988800 x^3")
print("13803759753640704000 x^2")
print("-8752948036761600000 x ")
print("2432902008176640000")
```

```
Part a:

x^20
-210 x^19
20615 x^18
-1256850 x^17
53327946 x^16
-1672280820 x^15
40171771630 x^14
-756111184500 x^13
11310276995381 x^12
-135585182899530 x^11
1307535010540395 x^10
-10142299865511450 x^9
63030812099294896 x^8
-311333643161390640 x^7
1206647803780373360 x^6
-3599979517947607200 x^5
8037811822645051776 x^4
-12870931245150988800 x^3
13803759753640704000 x^2
-8752948036761600000 x
2432902008176640000

 Part b:
```

In [18]:
```
import scipy
from scipy import optimize

print("\n Part b: \n")

n20 = float(1)
n19 = float(-210)
n18 = float(20615)
n17 = float(-1256850)
n16 = float(53327946 )
n15 = float(-1672280820)
n14 = float(40171771630)
n13 = float(-756111184500)
n12 = float(11310276995381)
n11 = float(-135585182899530)
n10 = float(1307535010540395)
n9 = float(-10142299865511450)
n8 = float(63030812099294896)
n7 = float(-311333643161390640)
n6 = float(1206647803780373360)
n5 = float(-3599979517947607200)
n4 = float(8037811822645051776)
n3 = float(-12870931245150988800)
n2 = float(13803759753640704000)
n1 = float(-8752948036761600000)
n0 = float(2432902008176640000)

n_values = [n0,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18,n19,n20]

def w(x):
    sum = 0
    for i in range(0,21):
        sum = sum + n_values[i]*x**i

    return sum

root = optimize.newton(w, 21, tol = 1.48e-06)
```

```
 Part b:

Root using Newton-Raphson method:
19.99995944008897
```

In [24]:
```
print("\n Part c: \n")

n20 = float(1)
n19 = float(-210)
n18 = float(20615)
n17 = float(-1256850)
n16 = float(53327946 )
n15 = float(-1672280820)
n14 = float(40171771630)
n13 = float(-756111184500)
n12 = float(11310276995381)
n11 = float(-135585182899530)
n10 = float(1307535010540395)
n9 = float(-10142299865511450)
n8 = float(63030812099294896)
n7 = float(-311333643161390640)
n6 = float(1206647803780373360)
n5 = float(-3599979517947607200)
n4 = float(8037811822645051776)
n3 = float(-12870931245150988800)
n2 = float(13803759753640704000)
n1 = float(-8752948036761600000)
n0 = float(2432902008176640000)

n_values = [n0,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18,n19,n20]

def w(x):
    sum = 0
    n_values[0] = 1 + 10**-8
    for i in range(0,21):
        sum = sum + n_values[i]*x**i

    return sum

root = optimize.newton(w, 21, tol = 1.48e-06)
print("Root with 10^-8 epsilon:")
print(root)
```

```
 Part c:

Root with 10^-8 epsilon:
21.00000094783766

 Root with 10^-6 epsilon:
21.00000094783766

 Root with 10^-4 epsilon:
21.00000094783766

 Root with 10^-2 epsilon:
21.00000094783766
```

In [ ]:

```

```