Homework 2

Name: Dylan Renard

Section: AMATH 301 A

Problem 1

(Make sure your code is somewhere)

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
y1 = 0
y2 = 0
y3 = 0
y4 = 0

# Initializing Coefficient terms
term12 = 0.1
term3 = 0.25
term4 = 0.5

for k in range(100000):
    y1 += term12
for k in range(100000000):
    y2 += term12
    y3 += term3
    y4 += term4
A6 = np.abs(10000 - y1)
A7 = np.abs(y2 - 10000000)
A8 = np.abs(25000000 - y3)
A9 = np.abs(y4 - 50000000)
```

Part a

In Problem 2 of the coding portion of the homework, I found the following values for x_1, x_2, x_3, and x_4.

x_1	x_2	x_3	x_4
1.8848368199542165e-08	0.018870549276471138	0.0	0.0

In [ ]:
```
q1 = {"x1":A6,"x2":A7,"x3":A8,"x4":A9}

q1 = dict(sorted(q1.items(), key=lambda item: item[1]))
```

```
print("Here are x1,x2,x3,x4 sorted from smallest to largest value")
print(q1)
```

```
Here are x1,x2,x3,x4 sorted from smallest to largest value
{'x3': 0.0, 'x4': 0.0, 'x1': 1.8848368199542165e-08, 'x2': 0.01887054927647
1138}
```

Part b

In [ ]:
```
# I think the reason X3 and X4 were shown as exactly zero whereas X1 and X2
# has to do with how floating point numbers are handled in programming langu
# Floating data types are 8 bits,
# so terms who are more easily represented by a factor of 8 will be more acc
# such as x3 and x4 who use 0.25 (1/4) and 0.5 (1/2) respectively,
# Whereas x1 and x2 use 0.1 which is harder to write as a fraction of 8 (~1/
# Both X1 and X2 use the addition term of 0.1 which might
```

Part c

In [ ]:
```
# X3 and X4 are exactly Zero.
# I guess this has to do with the fact
# that since they fit within a nice fraction of 8,
# [x3: 0.25] being 1/4 and [x4: 0.5] being 1/2
# there isn't any ambiguity when they are calculated by the computer?
```

Problem 2

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
xdata = np.linspace(-np.pi, np.pi, 100)
cosgraph = np.cos(xdata)

TaylorSwift1 = 0*xdata
for k in range(0,2):
    numerator = ((-1)**k)
    denominator = np.math.factorial(2*k)
    TaylorSwift1 += numerator*(xdata**(2*k))/denominator


TaylorSwift3 = 0*xdata
for k in range(0,4):
    numerator = ((-1)**k)
    denominator = np.math.factorial(2*k)
    TaylorSwift3 += numerator*(xdata**(2*k))/denominator


TaylorSwift14 = np.zeros(100)
for k in range(0,15):
    TaylorSwift14 += ((-1)**(k)/(np.math.factorial(2*k)))*(xdata**(2*k))

plt.rc('xtick',labelsize=10)
```

```
plt.rc('ytick', labelsize=10)
plt.plot(xdata, cosgraph, color = 'k', linewidth = 2, label = 'cos(x)')
plt.plot(xdata, TaylorSwift1, color = 'b', linestyle = '--', linewidth = 2,
plt.plot(xdata, TaylorSwift3, color = 'r', linestyle = '-.', linewidth = 2,
plt.plot(xdata, TaylorSwift14, color = 'm', linestyle = ':', linewidth = 2,
plt.xlabel("x-values", fontsize=15)
plt.ylabel("cos(x) Approximations", fontsize=15)
plt.title("cos(x) and its Taylor Approximations", fontsize=20)
plt.legend(fontsize=10)
plt.show()
```