HOMEWORK 2

Problem 1

a) In Python on a CS lab computer, make functions that have the exact below complexity:
   i) $50000000 \cdot n$
   ii) $50000000 \cdot n^2$
   iii) $50000000 \cdot 2^n$

b) Suppose you have a computer that can perform $50,000,000$ operations per second. For each of the functions, determine the largest input size n for which you can determine the result within 5 minutes.

c) Run each of the functions above, varying n from 1 to the number you determined in part b. Run those functions using the `energyusage` package on a Linux computer with an Intel processor (all of the lab computers meet these requirements and have the package installed already). The package will give you the information you'll need for the graphs in part d. An example run on a simple function f with input x and return value y is:

```
kwh, y = energyusage.evaluate(f, x, energyOutput=True)
```

where `kwh` is the resulting energy usage in kilowatt hours, and the time used will be printed to the command line output. Note that the CS lab machines perform *approximately* $50,000,000$ operations per second, so you can test your above calculations in this step.

d) Make two graphs for each of the functions above:
   i) a graph showing n on the x-axis versus time (in seconds) on the y-axis, and
   ii) a graph showing n on the x-axis versus energy usage (in kilowatt hours) on the y-axis.

   You may use whatever graphing software or package you prefer to create these graphs. Make sure you have properly labeled the axes and titled your graphs.

e) The power plants supplying electricity to your computer emit between 2.4 and 1.1 pounds of carbon dioxide per kWh. Which function minimizes the amount of carbon dioxide emitted for $n = 5$? What about over 5 minutes?

f) What do you notice about the graphs? Describe any conclusions you have reached about computational complexity and energy "complexity," along with any caveats to these conclusions.

Your handed in solution to this homework problem should include:

a) The functions you wrote.
b) The values of n you determined.
d) Two graphs for each function (a total of 6 graphs).
e) A sentence or two.
f) A short paragraph describing your conclusions.

Book Problem 2.7

The instructions are a bit confusing. In case you find it useful, here is additional explanation.

You should determine how long a "song script" would be if you had some clear way of expressing the repeats. You can think about this as the "encoding" of the full song. The length of this "encoding" is your $f(n)$.

For example, suppose the original lyrics have a total number of k verses. A (not very clever) "encoding" of the first few verses of "The Twelve Days of Christmas" looks like this (where individual verses are numbered, since they're too long to fit on a page):

```
1.  On the first day of Christmas my true love gave to me
     A partridge in a pear tree
2.  On the second day of Christmas my true love gave to me
     Two turtle doves and
     A partridge in a pear tree
3.  On the third day of Christmas my true love gave to me
     Three French hens
     Two turtle doves and
     A partridge in a pear tree
```

The pseudocode that generates the above encoding is:

```
Function 12daysofXmas()
    lines[1] = original lyrics verse 1
    ...
    lines[k] = original lyrics verse k
    for i = 1 to k do
        output verses[i]
    end
```

Confusingly, I've used a different definition of c below (the length of the longest verse) than the book did. In the below, c is not a constant. Sorry about adding this confusion!

Let the length of the longest verse be bounded by some variable c. Then the length of the original lyrics is at most kc. This encoding clearly has the same as number of lines as the original, plus the 3 additional lines used in the for loop. The initialization lines each have length $c + c_1$, where c_1 is a constant that denotes the length of "`lines[*] = `". Note $c_1 < c$. The lines in the for loop are shorter than the original lyric lines. Therefore the length of the encoding is $f(n) < (k(c + c_1) + 3c) = kc + kc_1 + 3c$, where $n = kc$. Thus $f(n)$ has linear growth.

However, you can of course encode this song so that the song sheet only has the following lines on it (as well as, perhaps, some short instructions):

```
1.  On the third day of Christmas my true love gave to me
2.  Three French hens
3.  Two turtle doves and
4.  A partridge in a pear tree
```

Please include a description of this encoding scheme and pseudo code. Focus on the length analysis. You may skip the proof of correctness.