(CS417/517) Machine Assignment 2

1 (5 points) Convert Decimal Mantissa to Arbitrary Mantissa

Write a program that receives a real number in decimal (base 10) and converts it into any base (e.g., 2, 8, 16, 60). This base must be accepted as the first command line argument.

- You must implement the algorithm discussed in Chapter 1.
- **You may not use libraries or built-in functions (e.g., Double.toHexString(...) in Java or "{0:b}".format(...) in Python)**
- *If you were careful in Machine Assignment 1, you should be able to modify your decimal to binary program.*

You may assume that all input is well-formed (i.e., all inputs are valid real numbers). You need not (and should not) expect illegal inputs (e.g., "0.1LOL").

1.1 Legal Input

Ideally, your program should handle all real numbers including: negative numbers, positive numbers, and those with non-zero integer components. However, you may (without penalty) restrict your expected input to numbers in the domain 0 to (inclusive) to 1 (exclusive)–i.e., $x \in [0, 1)$.

1.2 Sample Execution & Output

All input must be handled through command line arguments. Suppose you were implementing your solution in a Python 3.7 program, *convert_dec_to_any.py*. Program execution should be similar to:

```
./convert_dec_to_any.py 60 0.5 0.75 0.8 0.16666
```

The output should take a form similar to:

```
| Base 10 | Base 60 |
| :-------|:--------|
| 0.5     | 0.30    |
| 0.25    | 0.15    |
| 0.75    | 0.45    |
| 0.8     | 0.48    |
| 0.16666 | 0.10    |
```

Your conversions to base 60 may vary due to machine arithmetic (this definitely applies to the last row in the example).

```
./convert_dec_to_bin.py 2 0.5 0.25 0.75
```

The output should take a form similar to:

```
| Base 10 | Base 2 |
| :-------|:-------|
| 0.5     | 0.1    |
| 0.25    | 0.0;1  |
| 0.75    | 0.1;1  |
```

If you have a number that repeats, stop after *MAX_DIGITS*, which should be set as a global constant in your program. I suggest you start with *MAX_DIGITS = 8*.

2 (8 points) Approximating the Derivative

Write a program to compute an approximate value for the derivative of $f(x)$ using the finite difference formula

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{1}$$

Test your program using the function $\sin(x)$ for $x = 1$. The variable x will remain fixed (i.e., constant).

Determine the error by comparing your computed value with the built-in function $\cos(x)$. Loop over $h = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots \frac{1}{2^{30}}$

Your output should be similar to

```
|  h   |       x       | Approx. f'(x) |  Known f'(x)  |  Abs. Error   |
|:----:|--------------:|--------------:|--------------:|--------------:|
|2^-01 |    1.00000000 |    0.31204800 |    0.54030231 |    0.22825430 |
|2^-02 |    1.00000000 |    0.43005454 |    0.54030231 |    0.11024777 |
|2^-03 |    1.00000000 |    0.48637287 |    0.54030231 |    0.05392943 |
|2^-04 |    1.00000000 |    0.51366321 |    0.54030231 |    0.02663910 |
```

However, unlike this abbreviated example you must complete up to 2^{-30}.

Take the output of your program and plot h (x-axis) vs absolute error (y-axis). Set both axes (x-axis and y-axis) to logarithmic scales.

Is there a minimum value for the magnitude of the error? If such a value exists, how does it compare to $\sqrt{eps}$?