Floating-Point Numbers

Due at **4:00pm on Friday 20 September 2019**

What you need to get

- `YOU_a1q1.ipynb`: a jupyter notebook for Q1
- `YOU_a1q2.ipynb`: a jupyter notebook for Q2
- `YOU_a1q5.ipynb`: a jupyter notebook for Q5

If you want to typeset your solutions to Q3 and Q4 in LaTeX, then you're welcome to use:

- **https://www.overleaf.com/read/vztpkwrpmtcn**: (optional) Overleaf template

What you need to know

The notebook `YOU_a1q1` has a function called `dec2fp` that takes a numerical value as input and generates a binary floating-point representation of it. The inputs t, L and U specify a floating-point number system (FPNS), which we will denote $\mathcal{F}(\beta = 2, t, L, U)$, containing elements

$$b = \pm 0 \,.\, d_1\, d_2\, d_3 \;\ldots\; d_t \;\times\; 2^p \,,$$

where $d_k \in \{0, 1\}$, $d_1 \neq 0$, and $p \in \mathbb{Z}$ with $L \leq p \leq U$. If a value falls outside the range of values in the FPNS, then it returns an exception: `Inf`, `-Inf`, `NaN`, or 0 (for underflow). The value of zero is a special code in which the mantissa is all zeros and the exponent is zero.

The floating-point numbers will be stored as strings. For example,

- 0.1101×2^{-3} will be represented by the string '`+0.1101b-3`'
- -0.100010×2^{4} will be represented by the string '`-0.100010b4`'.

Note that the first character is always either a '+' or '-'. The number after the 'b' is the exponent for the base (the base is 2), although the exponent itself is represented in base-10. For example,

```
b = '+0.11100b4'
```

represents the number 0.11100×2^4, which has a value of 14. Hence,

```
b2 = dec2fp(14, 7, -20, 20)
```

returns the string '`+0.1110000b4`'. Type "`? dec2fp`" for more information.

You can perform arithmetic operations involving these binary strings using the function `fpMath` (also supplied in the notebook). The function takes two binary strings, a function, and t, L, and U. The output is another binary string. Note that functions in Python can be defined inline using the `lambda` notation. For example, the Python code

```
(lambda z1,z1: z1-z2)
```

returns a function that subtracts its second argument from its first argument. Thus, the call

```
fpMath(b1, b2, (lambda z1,z2: z1-z2), 3, -10, 10)
```

returns the binary code for the number that corresponds to `b1-b2`. Type "`? fpMath`" for more information.

What to do

1. **[4 marks]** Complete the Python function `randfp` in the `YOU_a1q1` notebook so that it randomly generates normalized binary floating-point numbers from the number system $\mathcal{F}(\beta = 2, t, L, U)$. Your function should work for values of t up to 52, and $-1022 \leq L < U \leq 1023$. You can read the function's documentation for more information (type "`? randfp`").

   Hint:
   To append strings in Python, simply 'add' strings. For example,

   ```
   b = 'hi' + ' there ' + str(15)
   ```

   will construct the string '`hi there 15`'.

2. **[4 marks]** Complete the function `fp2dec` (in the `YOU_a1q2` notebook) so that it converts binary floating-point numbers in $\mathcal{F}$ to their decimal equivalents. An incomplete version of the function is supplied as starter code. Its input is a string representing a binary floating-point number (as described in *What you need to know* above). It is sufficient to output an IEEE double-precision number as the decimal value.

   Hints:
   For this question, you might find the Python functions `find`, and `int` useful. Also, you can extract substrings using indexing. For example, if `b='+0.1001b3'`, then `b[2]` will return the string `'.'`, and `b[6:]` will return `'1b3'`. Furthermore, the Boolean expression `b[3]=='1'` would return a value of `True`. You **cannot**, however, use any other function that does the conversion for you. You must implement it yourself based on first principles.

3. **[4 marks]** Consider the normalized floating-point number system $\mathcal{F}(\beta = 7, t = 4, L = -8, U = 8)$, with elements of the form
   $$\pm 0.d_1 d_2 d_3 d_4 \times 7^p$$
   where $-8 \leq p \leq 8$ and $d_i \in \{0, \ldots, 6\}$. The number system is normalized, so $d_1 \neq 0$. The only exception is the zero element, in which all the mantissa digits are zero and the exponent is zero. For the following questions, state your answers in base-7.

   (a) What is the largest value in $\mathcal{F}$?

   (b) What is the value of $26530_7 \times 10000_7$ using this number system.

   (c) Derive machine epsilon for $\mathcal{F}$ from first principles (<u>not</u> using the general formula). In other words, what is the smallest value $E \in \mathcal{F}$ such that $fl(1 + E) > 1$.

   (d) What fraction of the values in $\mathcal{F}$ are smaller in magnitude than 1?

4. **[4 marks]** Let $\mathcal{F}$ be a floating-point number system with machine epsilon E, and suppose that a and b are numbers that may or may not be elements of $\mathcal{F}$. Show that the relative error for the expression $fl(a) \oplus fl(b)$ has the upper bound
   $$\frac{|(fl(a) \oplus fl(b)) - (a + b)|}{|a + b|} \leq \frac{|a| + |b|}{|a + b|} E(2 + E) \ .$$
   Justify each inequality that you introduce.

5. **[5 marks]** Consider the function
   $$F(x) = \frac{1}{1 - x} - \frac{1}{1 + x}$$

for $|x| < \frac{1}{2}$. The notebook `YOU_a1q5` contains the functions `F_exact` and `F_fp`. The function `F_exact` simply computes $F(x)$ for a given x using Python's default IEEE double-precision. We will refer to this version as "exact" because it is far more accurate than the alternatives that we will compare to in this question.

The function `F_fp` computes the same formula, but using the FPNS $\mathcal{F}(\beta, t, L, U) = (10, 4, -100, 100)$. Notice that it uses the function `fl` repeatedly, which is also included in the notebook. Notice also that all intermediate values are in the number system.

(a) For what range of x-values is it difficult to compute this expression accurately in floating-point arithmetic? Edit the notebook so that it creates a plot that compares `F_exact` to `F_fp`. Adjust the range on the x-axis so that it illustrates the inaccuracy of using $\mathcal{F}$ compared to the "exact" value. Plot x vs. F for both methods on the same axis (the plot should appear inline in the notebook).

(b) Algebraically rearrange the formula for $F(x)$ to get a new version so that the computation using $\mathcal{F}$ is more accurate than in part (a). Your derivation should be typeset in LaTeXin the notebook.

(c) Create a function called `F2_fp` that computes your new formula.

(d) Add code to the notebook that creates a second plot comparing `F_fp` and `F2_fp` over the same x-range as in (a). Be sure to add labels to the plot, as well as a legend, as in part (a).

What to submit

Rename each of your jupyter notebooks, replacing "`YOU`" with your WatIAM ID. For example, I would rename `YOU_a1q1.ipynb` to `jorchard_a1q1.ipynb`. Export each jupyter notebook as a PDF, and submit each PDF to Crowdmark. If you want, you can typeset your solutions to Q3 and Q4 in LaTeXor a Word document, write electronically (as on a tablet), or write it by hand and submit a high-quality photo or scan. It is your responsibility to ensure that your handwritten solutions are legible.