EX.0.3.5, Sauer

Do the following sums by hand in IEEE double precision computer arithmetic, using the Rounding to Nearest Rule. (Check your answers, using Matlab.)

a. $(1 + (2^{-51} + 2^{-53})) - 1$

b. $(1 + (2^{-51} + 2^{-52} + 2^{-53})) - 1$

EX.0.3.5, Sauer, solution, Langou

- Only turning the Python code is not a good answer.
- The copy-paste from this PDF to python code does not work great. It is better to copy-paste from colab.
- The Colab Jupyter Notebook is available at: `https://colab.research.google.com/drive/1Rp45zp-DmKzMdh9JO9Mn_X7KBiH34Bc`.

To check that we get the exact same floating-point number we will print the 64-bit representation of our floating-point numbers using the following Python function.

```
import struct
def double_to_hex(f):
  return hex(struct.unpack('<Q', struct.pack('<d', f))[0])
```

Do not forget to execute the piece of code above first.

a. $x_1 = (2^{-51} + 2^{-53})$ is a machine number and can be stored exactly so it is stored exactly. Then $x_2 = 1 + 2^{-51} + 2^{-53}$ is not a machine number so it cannot be stored exactly. We have to round either up or down. x_2 is in between the two machine numbers: $x_- = 1 + 2^{-51}$ and $x_+ = 1 + 2^{-51} + 2^{-52}$. We round to the nearest. Well it turns out that x_2 is right in the middle. So we need to take whichever number between x_- and x_+ that has for last mantissa bit a 0. (Note: x_- and x_+ are consecutive machine numbers, we are guaranteed that one of the two has a 0 for last mantissa and that the other has a 1. IEEE standard (arbitrarily) dictates to take the one with a 0.) So the question is which number has "no 2^{-52}". This is x_-. So x_2 is rounded to x_-. In other words $\mathrm{fl}(x_2) = x_-$. The next step is to subtract 1, so we expect to see 2^{-51} as an answer.

$$\boxed{\mathrm{fl}\left((1 + (2^{-51} + 2^{-53})) - 1\right) = 2^{-51}}$$

It might be better to see these numbers with bits. We have

$$\begin{aligned} x_- = 1 + 2^{-51} &= +2^0(1.0000000000000000000000000000000000000000000000000010)_2 \\ x_2 = 1 + 2^{-51} + 2^{-53} &= +2^0(1.0000000000000000000000000000000000000000000000000010|1)_2 \\ x_+ = 1 + 2^{-51} + 2^{-52} &= +2^0(1.0000000000000000000000000000000000000000000000000011)_2 \end{aligned}$$

Python Check: We check that

```
x = ( 1. + ( (2. ** (-51)) + (2. ** (-53)) ) ) - 1.
y = ( 2. ** (-51) )
print(x)
print(y)
print( double_to_hex(x) )
print( double_to_hex(y) )
```

```
4.440892098500626e-16
4.440892098500626e-16
0x3cc0000000000000
0x3cc0000000000000
```

b. $x_1 = 2^{-51} + 2^{-52} + 2^{-53}$ is a machine number and can be stored exactly so it is stored exactly. Then $x_2 = 1 + 2^{-51} + 2^{-52} + 2^{-53}$ is not a machine number so it cannot be stored exactly. We have to round either up or down. x_2 is in between the two machine numbers: $x_- = 1 + 2^{-51} + 2^{-52}$ and $x_+ = 1 + 2^{-50}$. We round to the nearest. Well it turns out that x_2 is right in the middle. So we need to take whichever number between x_- and x_+ that has for last mantissa bit a 0. So the question is which number has "no 2^{-52}". This is x_+. So x_2 is rounded to x_+. In other words $\text{fl}(x_2) = x_+$. The next step is to subtract 1, so we expect to see 2^{-50} as an answer.

$$\boxed{\text{fl}\left((1 + (2^{-51} + 2^{-52} + 2^{-53})) - 1\right) = 2^{-50}}$$

It might be better to see these numbers with bits. We have

$$\begin{aligned}
x_- = 1 + 2^{-51} + 2^{-52} &= +2^0(1.0000000000000000000000000000000000000000000000000011)_2 \\
x_2 = 1 + 2^{-51} + 2^{-52} + 2^{-53} &= +2^0(1.0000000000000000000000000000000000000000000000000011|1)_2 \\
x_+ = 1 + 2^{-50} &= +2^0(1.0000000000000000000000000000000000000000000000000100)_2
\end{aligned}$$

Python Check: We check that

```
x = ( 1. + ( (2. ** (-51)) + (2. ** (-52)) +  + (2. ** (-53)) ) ) - 1.
y = ( 2. ** (-50) )
print(x)
print(y)
print( double_to_hex(x) )
print( double_to_hex(y) )
```

```
4.440892098500626e-16
4.440892098500626e-16
0x3cc0000000000000
0x3cc0000000000000
```