Math 753/853 HW1: floating-point numbers, due Wed 9/14

Answer the questions with Julia code, Julia calculations, and words in comment strings (starting with #). You can also use Markdown cells for text, if you wish. Problem 0 is worked out for you as an example.

Problem 0. Use Julia's `nextfloat` function to determine the separation between 1.0 and the next floating point number (using the default `Float64` type).

In [16]:
```
nextfloat(1.0)-1.0
```

Out[16]:
```
2.220446049250313e-16
```

Can you explain the precise value of this number?

In [17]:
```
# Yes, 2.220446049250313e-16 is 2^-52, the next allowable value for a 52-bit mantissa
2.0^-52
```

Out[17]:
```
2.220446049250313e-16
```

Predict the next floating point number after 8.0 based on your understanding of floating-point numbers, and verify with `nextfloat`.

In [18]:
```
# 8 is represented as (1 + 0)*2^3. The next 64-bit float after 8.0 should be (1 + 2^-52)*2^3 == 8 + 2^-49.
8 + 2.0^-49
```

Out[18]:
```
8.000000000000002
```

In [19]:
```
nextfloat(8.0)
```

Out[19]:
```
8.000000000000002
```

In [21]:
```
# those look the same, but the representation of floats as decimal strings uses rounding
# let's check that the two values are exactly equal by subtracting
(8 + 2.0^-49) - nextfloat(8.0)
```

Out[21]:
```
0.0
```

Problem 1. Based on 9/7/2016 lecture material, what range of integers should a 16-bit integer type represent?

In [ ]:

Check your answer by running typemin(Int16) and typemax(Int16).

In [ ]:

Problem 2. Same as problem 1 for 32-bit integers.

In [ ]:

Problem 3. The standard 32-bit floating-point type uses 1 bit for sign, 8 bits for exponents, and 23 bits for the mantissa.

What is machine epsilon for a 32-bit float? Express your answer both as a power of two and a power of ten.

In [ ]:

How many digits of accuracy does the mantissa have?

In [ ]:

What are the minimum and maximum exponents, in base 10?

In [ ]:

Problem 4. The standard 16-bit floating-point type uses 1 bit for sign, 5 bits for exponents, and 10 bits for the mantissa. What size error do you expect in a 16-bit computation of 9.4 - 9 - 0.4?

In [ ]:

You can enter a 16-bit float in Julia with the syntax `9.4f0`. Compute 9.4 - 9 - 0.4 using 16-bit floats and verify your expectation.

In [ ]:

Problem 5. Find the roots of $x^2 - 4x + 6^{-28} = 0$ to several significant digits.

In [ ]:

Now that you know the answers, do you see a way you could have found them easily by a couple simple approximations?

Problem 6. Given x = 4778 + 3.77777e-10 and y = 4778 + 3.11111e-10, how many digits of accuracy do you expect in 64-bit calculations of the following?

x + y

x - y

x * y

x / y

Can you think of way to test your expectations in Julia? Note that evaluating the expressions in 64-bit arithmetic just gives you the 64-bit approximation, without telling you anythign about its accuracy.