Lily Davoren HW 1 09/11/2023

PROBLEM 1: FLOATING-POINT NUMBERS

a)

123456/2	0
61728/2	0
30864/2	0
15432/2	0
7716/2	0
3858/2	0
1929/2	1
964/2	0
482/2	0
241/2	1
120/2	0
60/2	0
30/2	0
15/2	1
7/2	1
3/2	1
1/2	1

Binary Representation: 11110001001000000_2

Hexadecimal: $1E240_{16}$

Explanation: If you divide a number by 2, the remainder can only be a 0 (for an even number) or a 1 (for an odd number), meaning that a binary representation can be reached by dividing a number and it subsequent quotients by 2 and keeping track of the remainder each time until the quotient is zero. Once all remainders have been found, the binary representation can be written by writing out all of the remainders starting from the last one and working up. This is because the last remainder represents the most significant bit, which has the greatest place value in the binary number. The hexadecimal representation is found by breaking up the binary representation into groups of four bits, beginning at the end. If the numerical representation of a binary group is greater than 9, then alphabetical representation is used starting at 10.

b) 11111111

c) 0 1 0011 0000001010

$\underline{2^4+2^1+2^0}$ $\underline{2^{-7}+2^{-9}}$

$16+2+1=19$ 0.009765625

biased exponent: $19-15=4$

$1.0000001010 \times 2^4 = (1+2^{-7}+2^{-9}) \times 16 = \boxed{16.15625}$

$1.0000001010 \times 2^4 = (1+\frac{1}{128}+\frac{1}{512}) \times 16$

$(\frac{512}{512}+\frac{4}{512}+\frac{1}{512}) \times 16$

$(\frac{517}{\cancel{512}_{32}}) \times \cancel{16} = \boxed{\frac{517}{32}}$

NEXT LARGEST: 0 1 00 11 0000001011

$19-15=4$ $2^{-7}+2^{-9}+2^{-10}$

$1.0000001011 \times 2^4 = (1+2^{-7}+2^{-9}+2^{-10}) \times 16 = \boxed{16.171875}$

Should be represented as 4 sig figs

d) There are 1024 or 2^{10} 16-bit floating-point numbers [16, 32) because there are 10 mantissa bits in a 16-bit number.

GAP SIZE: $\frac{1}{2^{10}} = \frac{1}{1024} \cdot 2^4 = 0.015625$

e) There are 1024 or 2^{10} 16-bit floating-point numbers [128, 256) because there are 10 mantissa bits in a 16-bit number.

GAP SIZE: $\frac{1}{2^{10}} = \frac{1}{1024} \cdot 2^7 = 0.125$

f) The code in floatingpoint_addition.py prints 128.5 instead of 128.4 and 128.8 instead of 128.6 in the second and third additions respectively because the precision of 16-bit floating-point numbers is limited, so the exact value 0.2 cannot be completely represented, so the value is rounded to the nearest value that can be represented in 16-bits. This rounding results in round-off error that accumulates every time the number is added. The third and fourth additions differ in that by order of operations the rounding error accumulates before adding it to the 128 in the fourth addition, so it has a smaller error than the third addition where the round-off error accumulates on the first value. The round-off error indicates that there is no representation between two numbers given a certain number of available bits.

PROBLEM 3:

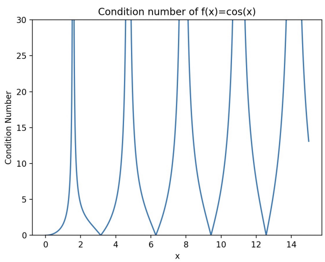
a)

```python
import numpy as np
import matplotlib.pyplot as plt

def condition(x):
    return abs(x*np.tan(x))

#extend x from 0 to 15
x = np.arange(0, 15, 0.01)

plt.plot(x, condition(x))
plt.ylim(0, 30)
plt.xlabel('x')
plt.ylabel('Condition Number')
plt.title('Condition number of f(x)=cos(x)')
plt.show()
```



The peaks that have been cutoff represent undefined values at multiples of $\pi/2$ because $x \cdot \frac{\sin(x)}{\cos(x)}$ when $\sin(x)=1$ and $\cos(x)=0$.

b) The function is most poorly-conditioned at (1.57, 1971.55197866), (4.71, 1971.548644699799), (7.85, 1971.5419767803721), (11.0, 2485.4593109961464), (14.14, 4991.058406720604)

c) $\pi/2 - 0.1$: 14.658904008200503

$3\pi/2 - 0.1$: 45.970040969253505

$5\pi/2 - 0.1$: 77.28117781030637

$7\pi/2 - 0.1$: 108.59231471135917

The condition number gets larger as the multiple of $\pi/2$ gets larger.

PROBLEM 2:

n	x^n	x^(n-1)	true error	approx error
1	2.000000	0.000000	0.264241	1.000000
2	2.500000	2.000000	0.080301	0.200000
3	2.666667	2.500000	0.018988	0.062500
4	2.708333	2.666667	0.003660	0.015385
5	2.716667	2.708333	0.000594	0.003067
6	2.718056	2.716667	0.000083	0.000511
7	2.718254	2.718056	0.000010	0.000073