[6 points] Perform the following calculations supposing that the binary values represent floating point numbers according to the IEEE 754 half-precision format (1 sign bit, 5-bit biased exponent, and 10-bit fraction). *Hint: check whether each binary value represents a special number before performing the calculations.*

(a) 0111110000000000 + 0111110000000000
(b) 0100010000000010 – 1011110000000100
(c) 0011110000000010 × 0001000000000100
(d) 1111111111111111 ÷ 0100011000000001

(a) 0111110000000000 + 0111110000000000

i) Check for special cases
→ Both numbers represent $+\infty$
→ Result is $+\infty$

Result = 0 11111 0000000000

(b) 0100010000000010 – 1011110000000100

i) Check for special cases
→ no special cases
ii) Transform subtraction to addition and negate second number
→ 0 10001 0000000010 + 0 01111 0000000100
iii) Align
→ Second number has smaller exponent
→ Add 2 to its exponent and shift its fraction to the right twice
→ Exponent of second number = 10001
→ Significand of second number = 0.0100000001
iv) Add significands (taking signs into consideration)
→ Significand of result = 1.0000000010 + 0.0100000001 = 1.0100000011
→ Sign bit of result = 0
v) Normalize
→ Significand is already normalized
→ Exponent of result = 10001
→ Fraction of result = 0100000011
vi) Round
→ Not needed
→ Fraction of result = 0100000011

Result = 0 10001 0100000011

(c) 0011110000000010 × 0001000000000100

i) Check for special cases
→ no special cases
ii) Add exponents (and subtract the bias)
→ Exponent of result = 01111 + 00100 − 01111 = 00100
iii) Calculate sign of result
→ Sign bit of result = 0
iv) Multiply significands
→ Significand of result = 1.0000000010 * 1.0000000100
= 00.00000001000000001000 + 01.00000000100000000000
= 01.00000001100000001000
v) Normalize
→ Significand is already normalized
→ Exponent of result = 00100
→ Fraction of result = 00000001100000001000

vi) Round
→ Suppose "rounding to nearest" is used
→ Fraction of result = 0000000110~~0000001000~~

Result = 0 00100 0000000110

(d) 1111111111111111 ÷ 0100011000000001

i) Check for special cases
→ First number is NaN
→ Result is NaN

Result = 1 11111 1111111111

(e) Perform the following calculations by interpreting the binary values according to the 12-bit floating-point format and rounding the results to the nearest representable number if needed:

i. 0 0000 0000000 ÷ 0 1111 0000000

ii. 0 1011 1000001 × 0 1001 1000000

i. 0 0000 0000000 ÷ 0 1111 0000000

* Check special cases
  → zero divided by ∞
  → result = zero = 0 0000 0000000

ii. 0 1011 1000001 × 0 1001 1000000

* Check special cases
  → no special case
* Add exponents (and subtract bias)
  → Exponent of result = 1011 + 1001 − 0111 = 1101
* Calculate sign of result
  → Sign of result = $\boxed{0}$ +ve * +ve
* Multiply significands

```
         1.1000001
         1.1000000
   ---------------
    11000001000000
   110000010000000
  ----------------
 10.01000011000000   product
```

★ Multiply significands

$$
\begin{array}{r}
1.1000001 \\
\times\ 1.1000000 \\
\hline
11000001000000 \\
11000001000000 \\
\hline
10.010000011000000
\end{array}
$$

product

★ Normalize

→ Shift product right one-bit position and increment exponent by 1

→ Product = 1.00100000110000000

→ Exponent = 1101 + 1 = $\boxed{1110}$

★ Round to nearest

→ Fraction = $001000\boxed{0}\underbrace{\cancel{11000000}}_{\text{greater than } 1000\ldots\ldots}$

$001000\boxed{1}$

Result = 0 1110 0010001

[9 points] Consider the IEEE 754 half-precision format in which floating point numbers are represented using 16 bits: 1 sign bit, 5-bit biased exponent, and 10-bit fraction.

(a) Convert the following numbers to their IEEE half-precision counterparts:

i. 109.6875

ii. -6.103515625E-5

(a)

9

i. $109.6875 = 1.713867 1875 * 2^6$

2

$\longrightarrow$ $\overset{15+6}{0\ 10101}\ 1011011011$

ii. $-6.103515625 * 10^{-5} = 1.0 * 2^{-14}$

2

$\longrightarrow$ 1 00001 0000000000