Eunki Chung
2272209

1. SP: 1.19209e-07
   DP: 2.22045e-16

2. $V = (-)^s M \cdot 2^E$

$E = e - bias$ where $e = e_0 \cdots e_{k-1}$, $bias = 2^{k-1} - 1$

$f = 1 + 0.f_{n-1} f_{n-2} \cdots f_0 = \frac{f_{n-1}}{2} + \frac{f_{n-2}}{2^2} + \cdots + \frac{f_0}{2^n}$

$$M = \begin{cases} 1 + f & \text{if } e \neq 0 \cdots 0. \\ f & \text{o.w.} \end{cases}$$

max E is when $e = 11 \cdots 10$ since $11 \cdots 1$ is reserved for inf.

$e = 11 \cdots 10 = 2^k - 2$

$E = 2^k - 2 - (2^{k-1} - 1)$

$= 2^k - 2 - 2^{k-1} + 1$

$= 2^{k-1} - 1$

M is maximized when $f_0 = \cdots = f_{n-1} = 1$

$M = 1 + f$

$= 1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^n}$

$= \frac{1 - \frac{1}{2^{n+1}}}{1 - \frac{1}{2}} = 2(1 - 2^{-n-1}) = 2 - 2^{-n}$

Largest normalized : $V_{max} = (-)^0 \cdot (2 - 2^{-n}) \cdot 2^{(2^{k-1} - 1)}$

The smallest normalized number is just negative largest number, V_{max}.

i.e., $V_{min} = -V_{max}$

Using the above formula,

i) SP (32 bit) ; $k = 8$, $n = 23$

Largest number:

$$V_{max} = (2 - 2^{-n}) \cdot 2^{(2^{k-1} - 1)}$$

$$= (2 - 2^{-23}) \cdot 2^{(2^{8-1} - 1)}$$

$$= (2 - 2^{-23}) \cdot 2^{127}$$

$$= 2^{128} - 2^{104}$$

$$= 2^{104}(2^{24} - 1)$$

Smallest number:

$$V_{min} = -2^{104}(2^{24} - 1)$$

$$\begin{cases} \text{Largest}: 2^{104}(2^{24} - 1) = 3.40282 \times 10^{38} \\ \text{Smallest}: -2^{104}(2^{24} - 1) = -3.40282 \times 10^{38} \end{cases}$$

ii) DP (64 bit) ; $k = 11$, $n = 52$

Largest number:

$$V_{max} = (2 - 2^{-n}) \cdot 2^{(2^{k-1} - 1)}$$

$$= (2 - 2^{-52}) \cdot 2^{(2^{11-1} - 1)}$$

$$= (2 - 2^{-52}) \cdot 2^{1023}$$

$$= 2^{1024} - 2^{971}$$

$$= 2^{971}(2^{53} - 1)$$

Smallest number:

$$V_{min} = -2^{971}(2^{53} - 1)$$

$$\begin{cases} \text{Largest}: 2^{971}(2^{53} - 1) = 1.79769 \times 10^{308} \\ \text{Smallest}: -2^{971}(2^{53} - 1) = 1.79769 \times 10^{308} \end{cases}$$

3. 200 * 300 * 400 * 500 = -884901888

In IEEE 754, Overflow and Underflow is types of exceptions for FP operation

Overflow occurs when the correctly rounded result of an operation is larger in magnitude than the largest representable finite number. That is, Overflow occurs when the exponent value E is greater than exponent upper bound U.

For IEEE SP, Overflow occurs if $E > 127$.
For IEEE DP, Overflow occurs if $E > 1023$

Underflow occurs when either the exact result or correctly rounded result is smaller in magnitude than the smallest representable normalized number.

For IEEE SP, Underflow occurs if $V < 2^{-126}$
For IEEE DP, Underflow occurs if $V < 2^{-1022}$

4. # of normalized floating numbers.

$$2-2^{k-1} \leq E \leq 2^{k-1}-1, \quad \forall E \in \mathbb{Z}$$

$$2^{k-1}-1-2+2^{k-1}+1 = 2^k-2$$

M can have 2^n different values.

$$\therefore \# = \underbrace{2}_{\text{sign}} \times \underbrace{2^n}_{\text{frac}} \times \underbrace{(2^k-2)}_{\text{exp.}} = 2^{n+1} \cdot (2^k-2)$$

i) SP

$$2^{24} \times (2^8-2)$$

ii) DP

$$2^{53}(2^{11}-2)$$

5. Consider a 6-bit FP System.

$$d = \underbrace{\boxed{d_5}}_{s=1}\underbrace{\boxed{d_4 \mid d_3 \mid d_2}}_{k=3}\underbrace{\boxed{d_1 \mid d_0}}_{n=2}$$

i) $d_5 = 0$

	base 2		base 10
$e =$	$d_4\ d_3\ d_{2\,2}$	$=$	$e_2 e_1 e_0$
	0 0 0	$\longrightarrow$	Denormalized
	0 0 1	$\longrightarrow$	1
	0 1 0	$\longrightarrow$	2
	0 1 1	$\longrightarrow$	3
	1 0 0	$\longrightarrow$	4
	1 0 1	$\longrightarrow$	5
	1 1 0	$\longrightarrow$	6
	1 1 1	$\longrightarrow$	inf.

$E = e_2 e_1 e_{0\,2} - \text{bias}$ where $\text{bias} = 2^{k-1} - 1 = 2^{3-1} - 1 = 3$

$e_2\ e_1\ e_0$	E
0 0 1	-2
0 1 0	-1
0 1 1	0
1 0 0	1
1 0 1	2
1 1 0	3

$d_1\ d_0$	f	$M = 1 + f$
0 0	$0 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2^2} = 0$	1
0 1	$0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2^2} = \frac{1}{4}$	$\frac{5}{4}$
1 0	$1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2^2} = \frac{1}{2}$	$\frac{3}{2}$
1 1	$1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2^2} = \frac{3}{4}$	$\frac{7}{4}$

Normalized numbers $\qquad V = (-)^s M 2^E$

S_0	M_{00}	E_{001}	$1 \cdot 2^{-2}$	$= \frac{1}{4}$
S_0	M_{00}	E_{010}	$1 \cdot 2^{-1}$	$= \frac{1}{2}$
S_0	M_{00}	E_{011}	$1 \cdot 2^{0}$	$= 1$
S_0	M_{00}	E_{100}	$1 \cdot 2^{1}$	$= 2$
S_0	M_{00}	E_{101}	$1 \cdot 2^{2}$	$= 4$
S_0	M_{00}	E_{110}	$1 \cdot 2^{3}$	$= 8$
S_0	M_{01}	E_{001}	$\frac{5}{4} \cdot 2^{-2}$	$= \frac{5}{16}$
S_0	M_{01}	E_{010}	$\frac{5}{4} \cdot 2^{-1}$	$= \frac{5}{8}$
S_0	M_{01}	E_{011}	$\frac{5}{4} \cdot 2^{0}$	$= \frac{5}{4}$
S_0	M_{01}	E_{100}	$\frac{5}{4} \cdot 2^{1}$	$= \frac{5}{2}$
S_0	M_{01}	E_{101}	$\frac{5}{4} \cdot 2^{2}$	$= 5$
S_0	M_{01}	E_{110}	$\frac{5}{4} \cdot 2^{3}$	$= 10$
S_0	M_{10}	E_{001}	$\frac{3}{2} \cdot 2^{-2}$	$= \frac{3}{8}$
S_0	M_{10}	E_{010}	$\frac{3}{2} \cdot 2^{-1}$	$= \frac{3}{4}$
S_0	M_{10}	E_{011}	$\frac{3}{2} \cdot 2^{0}$	$= \frac{3}{2}$
S_0	M_{10}	E_{100}	$\frac{3}{2} \cdot 2^{1}$	$= 3$
S_0	M_{10}	E_{101}	$\frac{3}{2} \cdot 2^{2}$	$= 6$
S_0	M_{10}	E_{110}	$\frac{3}{2} \cdot 2^{3}$	$= 12$
S_0	M_{11}	E_{001}	$\frac{7}{4} \cdot 2^{-2}$	$= \frac{7}{16}$
S_0	M_{11}	E_{010}	$\frac{7}{4} \cdot 2^{-1}$	$= \frac{7}{8}$
S_0	M_{11}	E_{011}	$\frac{7}{4} \cdot 2^{0}$	$= \frac{7}{4}$
S_0	M_{11}	E_{100}	$\frac{7}{4} \cdot 2^{1}$	$= \frac{7}{2}$
S_0	M_{11}	E_{101}	$\frac{7}{4} \cdot 2^{2}$	$= 7$
S_0	M_{11}	E_{110}	$\frac{7}{4} \cdot 2^{3}$	$= 14$

Denormalized numbers, $E_{000} = 1 - \text{bias} = 1 - 3 = -2$

$V = (-)^s f 2^E$

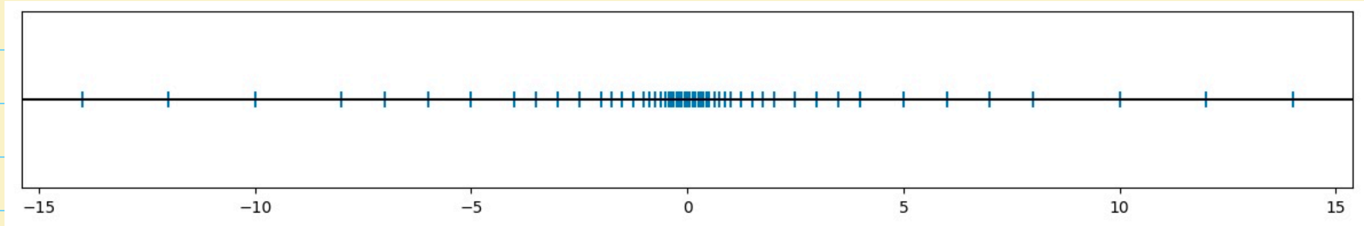
S_0	f_{00}	E_{000}	$0 \cdot 2^{-2}$	$= 0$
S_0	f_{01}	E_{000}	$\frac{1}{4} \cdot 2^{-2}$	$= \frac{1}{16}$
S_0	f_{10}	E_{000}	$\frac{1}{2} \cdot 2^{-2}$	$= \frac{1}{8}$
S_0	f_{11}	E_{000}	$\frac{3}{4} \cdot 2^{-2}$	$= \frac{3}{16}$

Since we have $s=1$ case too, all representable numbers are the set of all numbers listed above and the corresponding negative numbers.

The full list of representable numbers is

[-14.0, -12.0, -10.0, -8.0, -7.0, -6.0, -5.0, -4.0, -3.5, -3.0, -2.5, -2.0, -1.75, -1.5, -1.25, -1.0, -0.875, -0.75, -0.625, -0.5, -0.4375, -0.375, -0.3125, -0.25, -0.1875, -0.125, -0.0625, -0.0, 0.0, 0.0625, 0.125, 0.1875, 0.25, 0.3125, 0.375, 0.4375, 0.5, 0.625, 0.75, 0.875, 1.0, 1.25, 1.5, 1.75, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0, 8.0, 10.0, 12.0, 14.0]

If we plot these numbers on a line we get,



Appendix

Source code for the plot in problem 5

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

exp = np.arange(-2, 4, dtype=float)

mantissa = np.array([1., 5/4, 3/2, 7/4])

values = []
for M in mantissa:
    values.append(M * np.power(2, exp))

normalized = np.concatenate(values)
normalized
```

```
array([ 0.25  ,  0.5   ,  1.    ,  2.    ,  4.    ,  8.    ,  0.3125,
        0.625 ,  1.25  ,  2.5   ,  5.    , 10.    ,  0.375 ,  0.75  ,
        1.5   ,  3.    ,  6.    , 12.    ,  0.4375,  0.875 ,  1.75  ,
        3.5   ,  7.    , 14.    ])
```

```
f = mantissa - 1
denormalized = f * (2 ** -2)
denormalized
```

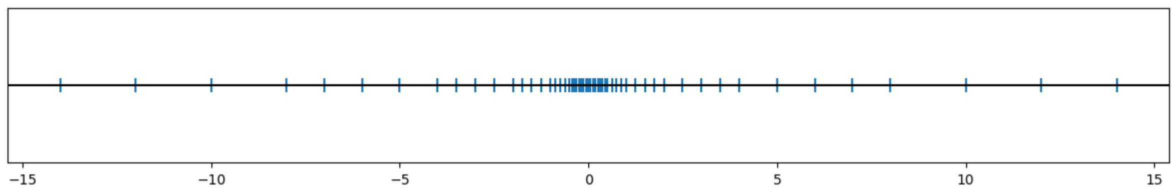
```
array([0.    , 0.0625, 0.125 , 0.1875])
```

```
numbers = np.concatenate([normalized, denormalized])
numbers.sort()
numbers = np.concatenate([np.flip(-numbers), numbers])
numbers
```

```
array([-14.    , -12.    , -10.    ,  -8.    ,  -7.    ,  -6.    ,
        -5.    ,  -4.    ,  -3.5   ,  -3.    ,  -2.5   ,  -2.    ,
        -1.75  ,  -1.5   ,  -1.25  ,  -1.    ,  -0.875 ,  -0.75  ,
        -0.625 ,  -0.5   ,  -0.4375,  -0.375 ,  -0.3125,  -0.25  ,
```

```
-0.1875,  -0.125 ,  -0.0625,  -0.     ,   0.     ,   0.0625,
 0.125 ,   0.1875,   0.25  ,   0.3125,   0.375 ,   0.4375,
 0.5   ,   0.625 ,   0.75  ,   0.875 ,   1.     ,   1.25  ,
 1.5   ,   1.75  ,   2.    ,   2.5   ,   3.     ,   3.5   ,
 4.    ,   5.    ,   6.    ,   7.    ,   8.     ,  10.    ,
12.    ,  14.    ])
```

```python
plt.figure(figsize=(15, 2))
plt.scatter(x = numbers, y = np.zeros_like(numbers), s=100, marker="|")
plt.yticks([])
plt.axhline(0, color="black")
plt.show()
```



Verification of Overflow and Underflow in SP and DP

Here, I used the estimated machine epsilon from problem 1.

$$\epsilon_{SP} = 1.19209\text{e-}07$$

$$\epsilon_{DP} = 2.22045\text{e-}16$$

```cpp
#include <iostream>
#include <cmath>

int main()
{
    float Vf = std::powf(2, 104) * (std::powf(2, 24) - 1);
    float vf = -Vf;
    float nvf = std::powf(2, -126);
    float epsf = 1.19209e-07f;

    std::cout << "Largest float V = " << Vf << std::endl;
```

```
    std::cout << "Smallest float v = " << vf << "\n\n";

    std::cout << "Overflow\n";

    std::cout << "V + (2^102 * (2 - eps)) = "
              << Vf + (std::powf(2, 102) * (2.0f - epsf)) << "\n\n";
    std::cout << "V + 2^103 = " << Vf + (std::powf(2, 102) * 2.0f) << "\n\n";

    std::cout << "v - (2^102 * (2 - eps)) = "
              << vf - (std::powf(2, 102) * (2.0f - epsf)) << "\n\n";
    std::cout << "v - 2^103 = " << vf - (std::powf(2, 102) * 2.0f) << "\n\n";

    std::cout << "Underflow\n";

    std::cout << "Smallest normal float = " << nvf << "\n\n";
    std::cout << "Smallest normal float - eps = " << nvf - epsf << "\n\n";

    double Vd = std::pow(2, 971) * (std::pow(2, 53) - 1);
    double vd = -Vd;
    double nvd = std::pow(2, -1022);
    double epsd = 2.22045e-16;

    std::cout << "Largest double: " << Vd << std::endl;
    std::cout << "Smallest double: " << vd << "\n\n";

    std::cout << "Overflow\n";
    std::cout << "V + (2^969 * (2 - eps)) = "
              << Vd + (std::pow(2, 969) * (2.0 - epsd)) << "\n\n";
    std::cout << "V + 2^970 = " << Vd + (std::pow(2, 969) * 2.0) << "\n\n";

    std::cout << "v - (2^969 * (2 - eps)) = "
              << vd - (std::pow(2, 969) * (2.0 - epsd)) << "\n\n";
    std::cout << "v - 2^970 = " << vd - (std::pow(2, 969) * 2.0) << "\n\n";

    std::cout << "Underflow\n";
    std::cout << "Smallest normal double = " << nvd << "\n\n";
    std::cout << "Smallest normal double - eps = " << nvd - epsd << "\n\n";

    return 0;
}
```

Output

```
Largest float V = 3.40282e+38
Smallest float v = -3.40282e+38

Overflow
V + (2^102 * (2 - eps)) = 3.40282e+38

V + 2^103 = inf

v - (2^102 * (2 - eps)) = -3.40282e+38

v - 2^103 = -inf

Underflow
Smallest positive normal float = 1.17549e-38

Smallest positive normal float - eps = -1.19209e-07

Largest double: 1.79769e+308
Smallest double: -1.79769e+308

Overflow
V + (2^969 * (2 - eps)) = 1.79769e+308

V + 2^970 = inf

v - (2^969 * (2 - eps)) = -1.79769e+308

v - 2^970 = -inf

Underflow
Smallest positive normal double = 2.22507e-308

Smallest positive normal double - eps = -2.22045e-16
```