Question 1

Special floating-point numbers

- If floating point numbers are all normalized, the spacing between 0 and $b^{e_{\min}}$ is just $b^{e_{\min}}$, whereas the spacing between $b^{e_{\min}}$ and $b^{e_{\min}+1}$ is $b^{e_{\min}-p+1}$. With subnormal numbers, the spacing between 0 and $b^{e_{\min}}$ can be $b^{e_{\min}-p}$, which is more consistent with the spacing just above $b^{e_{\min}}$.

그런데 Spacing between 0 and $b^{e_{min}}$ with subnormal numbers는

$$0|00\ldots000|000\ldots001 \;-\; 0|00\ldots000|000\ldots000 = 0|00\ldots001|000\ldots000 \;-\; 0|00\ldots000|111\ldots111 = b^{e_{min}} \times b^{-(p-1)} = b^{e_{min}-p+1}$$

이므로 $b^{e_{\min}-p}$ 가 아니라 $b^{e_{min}-p+1}$가 되어야 하는 것 아닌가요?

Question 2

Machine precision

- If a positive real number $x \in \mathbb{R}$ is represented by $[x]$ in the floating point arithmetic, then

$$[x] = \left(1 + \sum_{i=1}^{p-1} \frac{d_{i+1}}{2^i}\right) \times 2^e.$$

Thus $x - \frac{2^e}{2^p} < [x] \le x + \frac{2^e}{2^p}$

그런데 다음과 같은 반례가 있기 때문에 마지막 부등식이 $<$ 가 아닌 $\le$ 가 되어야 한다고 생각합니다.
Let $\;x = 1 + \frac{1}{2^p} \Rightarrow [x] = 1 = 1.00\ldots00 \times 2^0 \quad \because \frac{1}{2^p} < \varepsilon_{\max}$
Hence $\;x - \frac{1}{2^p} = x - \frac{2^e}{2^p} = [x]$ where $e = 0$ in this case.

Question 3

Catastrophic cancellation

- **Scenario 2** (catastrophic cancellation): Subtraction of two nearly equal numbers eliminates significant digits. $a - b$ where $a \approx b$.

Analysis: Let

$$[x] = 1 + \sum_{i=1}^{p-2} \frac{d_{i+1}}{2^i} + \frac{1}{2^{p-1}}, \quad [y] = 1 + \sum_{i=1}^{p-2} \frac{d_{i+1}}{2^i} + \frac{0}{2^{p-1}}.$$

- $[x] - [y] = \frac{1}{2^{p-1}} = [[x] - [y]]$.
- The true difference $x - y$ may lie anywhere in $(0, \frac{1}{2^{p-2}} + \frac{1}{2^{2p}}]$.

그런데 마지막 문장에서 $x - y$ 의 범위가 $(0, \frac{1}{2^{p-2}} + \frac{1}{2^{2p}}]$이 아니라 $(0, \frac{1}{2^{p-2}}]$가 되어야 한다고 생각합니다. 그 이유는 다음과 같습니다.
In the machine precision section, we proved $x - \frac{2^e}{2^p} \le [x] \le x + \frac{2^e}{2^p}$ for a positive $x \in \mathbb{R}$.
In this case, $e = 0$ so $[x] - \frac{1}{2^p} \le x \le [x] + \frac{1}{2^p}$ and $[y] - \frac{1}{2^p} \le y \le [y] + \frac{1}{2^p}$ where $[x] - \frac{1}{2^p} = [y] + \frac{1}{2^p}$
Since $[x] > [y]$, the strict inequality $x > y$ must be true. $x - y > 0$. But there is no positive lower bound for $x - y$.
An upper bound of $x - y$ is $\frac{1}{2^{p-2}}$ $(\because x - y \le [x] + \frac{1}{2^p} - \left([y] - \frac{1}{2^p}\right) = [x] - [y] + \frac{1}{2^{p-1}} = \frac{1}{2^{p-1}} + \frac{1}{2^{p-1}} = \frac{1}{2^{p-2}})$

Question 4

Overflow and underflow of floating-point number

In [1]:

```julia
@show bitstring(floatmax(Float64))
@show bitstring( 2 - 2^(-52))
@show bitstring((2 - 2^(-52)) * (2^10))
@show bitstring((2 - 2^(-52)) * (2^50))
@show bitstring((2 - 2^(-52)) * (2^100))
@show bitstring((2 - 2^(-52)) * (2^1023))
```

```
bitstring(floatmax(Float64)) = "0111111111101111111111111111111111111111111111111111111111111111"
bitstring(2 - 2 ^ -52) = "0011111111111111111111111111111111111111111111111111111111111111"
bitstring((2 - 2 ^ -52) * 2 ^ 10) = "0100000010011111111111111111111111111111111111111111111111111111"
bitstring((2 - 2 ^ -52) * 2 ^ 50) = "0100001100011111111111111111111111111111111111111111111111111111"
bitstring((2 - 2 ^ -52) * 2 ^ 100) = "0000000000000000000000000000000000000000000000000000000000000000"
bitstring((2 - 2 ^ -52) * 2 ^ 1023) = "0000000000000000000000000000000000000000000000000000000000000000"
"0000000000000000000000000000000000000000000000000000000000000000"
```

Normalized number representation 으로 나타낼 수 있는 가장 작은 숫자와 가장 큰 숫자가 어떤 숫자인지 알아보고자 위와 같은 코드를 실행했습니다. 최소 숫자는 $2^{e_{\min}} \times 1.000\ldots00 = 2^{-1022}$였으며 문제 없이 원하는 결과가 나왔습니다.
최대 숫자는 bitstring으로 확인한 바에 의하면 $2^{e_{\max}} \times 1.111\ldots11 = 2^{1023} \times (1 + 2^{-1} + 2^{-2} + \cdots + 2^{-(p-1)}) = 2^{1023} \times (2 - 2^{-(p-1)}) = 2^{1023} \times (2 - 2^{52})$ 였으며 이를 계산해 보고자 $(2 - 2^{-52})$에다가 2^e 를 곱해보았는데 e 값이 10이나 50일 때는 정상적으로 계산이 되는데 100이나 $e_{\max} = 1023$등의 값을 넣으면 결과가 0이 되어버리는 비정상적인 결과가 나옵니다. 이런 오류가 발생하는 이유가 무엇인지 궁금합니다.