5. More Primitive Recursion over nat (10 Points)

Specify and implement the following functions:

- \( \text{half} : \text{nat} \rightarrow \text{nat} \) with the same semantics as in the lecture. But this time use an auxiliary function \( \text{half}' : \text{nat} \rightarrow \alpha \rightarrow \text{nat} \) with an extra argument of a suitable type \( \alpha \).

- \( \text{div} : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat} \). \( \text{div} \ n \ m \) computes the quotient \( n/(m + 1) \).

- \( \text{log} : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat} \). \( \text{log} \ n \ b \) computes the logarithm of \( n + 1 \) to the base \( b + 2 \).

6. More Primitive Recursion over list (10 Points)

Specify and implement the following functions:

- \( \text{flat} : \tau \text{list list} \rightarrow \tau \text{list} \). The function \( \text{flat} \) gets a list of lists \( ll \) and “flattens” it into a simple list, by concatenating all lists contained in \( ll \). This is to be implemented without the use of \textit{append} and by looking at every element only once.

- \( \text{merge} : \tau \text{list} \rightarrow \tau \text{list} \rightarrow \tau \text{list} \). \( \text{merge} \ x\text{s} \ y\text{s} \) takes the first element of \( x\text{s} \), then the first of \( y\text{s} \), then the second of \( x\text{s} \) etc. and puts them into a joint list. If one list is empty already, the remainder of the other list is simply appended.

7. Binary Representation of Natural Numbers (10 Points)

Introduce a new inductive datatype \( \text{bnat} \) for a binary representation of the natural numbers and specify and implement the successor function \( \text{bsucc} : \text{bnat} \rightarrow \text{bnat} \) and addition \( \text{badd} : \text{bnat} \rightarrow \text{bnat} \rightarrow \text{bnat} \).

Good luck!