1. Use one of the `List.fold` functions to implement `maximum : int list -> int`, which returns the highest value in a list. Make sure to consider all cases!

2. A **Gorn address** specifies how to get to a node in a tree starting from the root. It is an integer list whose elements identify which child to enter when traveling downwards. For this problem, take the first child to be number 1. For instance, in this tree

\[ \text{S} \]

\[ \text{Name} \]

\[ \text{John} \]

\[ \text{VP} \]

\[ \text{V} \]

\[ \text{gave} \]

\[ \text{NP} \]

\[ \text{D} \]

\[ \text{the} \]

\[ \text{N} \]

\[ \text{book} \]

\[ \text{PP} \]

\[ \text{P} \]

\[ \text{Name} \]

\[ \text{to} \]

\[ \text{Mary} \]

the Gorn address of the node P is \([2;3;1]\). The leaf node labelled *John* is addressed as \([1;1]\). The root has the empty address \([\ ]\) i.e. the path taken to get to the root from the root has no steps.

Define a type `tree` for labeled trees with an arbitrary number of branches. Write functions for these operations on values of type `tree`.

- `create a` returns a one-node tree with label `a`
- `build a l` returns a tree with root labeled `a` and subtrees given by the list `l`
- `at tree g` returns subtree of `tree` at Gorn address `g`, raising the exception `Missing` if there is no such subtree

Demonstrate your functions on some example trees.

Optional (Bonus): See if you can write a question-formation transformation that maps “John was washing dishes” to “Was John washing dishes.” What English-specific knowledge is used to implement this transformation?