User-defined data types

Multiple choice: The expression

```
Node (Leaf 1) (Leaf 2)
```

is a value of the data type:

- i)

  ```
  data Tree = Node | Leaf | Int
  ```

- ii)

  ```
  data Tree = Leaf Int | Node Int Int
  ```

- iii)

  ```
  data Tree = Leaf Tree | Node Int Int
  ```

- iv)

  ```
  data Tree = Leaf Int | Node Tree Tree
  ```

- v)

  ```
  data Tree = Leaf Tree | Node Tree Tree
  ```

Trees

Given the data type declaration

```
data Tree = Leaf Int | Node Int Tree Tree
```

write a function dfs :: Tree -> [Int] to list all integers in a tree in a depth-first, left-to-right traversal.

Trees

Given the data type declaration

```
data Tree = Leaf Int | Node Int Tree Tree
```

write a function bfs :: Tree -> [Int] to list all integers in a tree in a breadth-first, left-to-right traversal.

Concepts

What is a polymorphic function? Give an example and a counterexample.

Concepts

What is an overloaded function? Give an example and a counterexample.

Concepts

What is a curried function? Give an example and a counterexample.

Concepts

What is a higher-order function? Give an example and a counterexample.

Evaluation

Given the declarations

```
data T = A | B T T
f A = 1
f (B t' t'') = f t' + f t'' + 1
```

what does the expression

```
f (B (B A A) (B A A))
```

evaluate to?

Evaluation

Given the declarations

```
data T = A Char | B [T]
f (A c) = [c]
f (B ts) = concatMap f ts
```

what does the expression

```
f (B [B [A 'r', A 'o'], B [A 's'], B [A 'e']])
```

evaluate to?

Types and type classes

What is the most general type of the expression

```
[reverse, tail, take 3]
```

?

Types and type classes

What is the most general type of the expression

```
[reverse, tail, take 3, zipWith (&&) [True ..]]
```

?

Types and type classes

What is the most general type of the expression

```
[reverse, tail, take 3, filter even]
```

?

Types and type classes

What is the most general type of the expression

```
[reverse, tail, take 3, zipWith (:) [0..]]
```

?

Types and type classes

What is the most general type of the expression

```
[reverse, tail, take 3, zipWith (+) [0..]]
```

?

Types and type classes

What is the most general type of the expression

```
[reverse, tail, take 3, zipWith (:) ['a'..]]
```

?

Laws

Prove that for all finite lists xs,

```
foldr f e xs = foldl (flip f) e (reverse xs)
```