Assignment 1

Deadline for solutions: 04.05.2023

Exercise 1 Shortest Math Paper (5 Points)

The shortest math paper in history is allegedly the following one:

> **COUNTEREXAMPLE TO EULER'S CONJECTURE ON SUMS OF LIKE POWERS**
>
> BY L. J. LANDER AND T. R. PARKIN
>
> Communicated by J. D. Swift, June 27, 1966
>
> A direct search on the CDC 6600 yielded
>
> $$27^5 + 84^5 + 110^5 + 133^5 = 144^5$$
>
> as the smallest instance in which four fifth powers sum to a fifth power. This is a counterexample to a conjecture by Euler [**1**] that at least n nth powers are required to sum to an nth power, $n > 2$.
>
> REFERENCE
>
> **1.** L. E. Dickson, *History of the theory of numbers*, Vol. 2, Chelsea, New York, 1952, p. 648.

The above-mentioned counterexample is apparently just a four-tuple (a, b, c, d) of natural numbers (one can obviously assume that $a \leqslant b \leqslant c \leqslant d$), such that $\sqrt[5]{a^5 + b^5 + c^5 + d^5}$ is a natural number again. Write a Haskell program that brute force finds the above solution.

Hint: Use laziness and list comprehension. Depending on your machine capacities, search can take some minutes.

Exercise 2 Reasoning about Sorted Trees (7 Points)

Consider the following implementation of binary trees is Haskell:

```haskell
data BTree a = Leaf a | Branch (BTree a) (BTree a)
    deriving (Eq, Show)

btMin :: Ord a => BTree a -> a
btMin (Leaf l)       = l
btMin (Branch t s)   = min (btMin t) (btMin s)

btMax :: Ord a => BTree a -> a
```

```
btMax (Leaf l)        = l
btMax (Branch t s)    = max (btMax t) (btMax s)

isSorted :: Ord a => BTree a -> Bool
isSorted (Leaf y)       = True
isSorted (Branch t s) = isSorted t && isSorted s && btMax t <= btMin s

insert :: Ord a => a -> BTree a -> BTree a

insert x (Leaf y) | x <= y    = Branch (Leaf x) (Leaf y)
insert x (Leaf y) | otherwise = Branch (Leaf y) (Leaf x)

insert x (Branch t s) | x <= btMax t = Branch (insert x t) s
insert x (Branch t s) | otherwise    = Branch t (insert x s)
```

where

- `BTree a` is a type of binary trees with terminal nodes in `a`;
- `btMin` and `btMax` compute the least and the greatest element of a given tree;
- `isSorted` checks if a tree is sorted;
- `insert` inserts a new element to a tree.

(a) Provide an example of a *pre-order* `a`, and such elements `t` and `x` that `isSorted t`, but not `isSorted (insert x t)`. Recall that `a` is a pre-order if it satisfies `x <= x` (reflexivity), and `x <= y` with `y <= z` jointly imply `x <= z` (transitivity).

Hint: You need to exploit the fact that `a` need not be a *total pre-order*, i.e. it need not be the case that for any two elements `x` and `y`, `x <= y` or `y <= x`.

(b) Prove that if `x` is a total pre-order, `isSorted t` does entail `isSorted (insert x t)`.

Hint: Consider induction on `t`; consider strengthening the claim as follows: instead of showing `isSorted (insert x t)`, show that `isSorted (insert x t) && (btMax (insert x t) <= max (btMax t) x) && (btMin (insert x t) =< min (btMin t) x)`.

Exercise 3 Getting Real (8 Points)

Consider a notion of number which includes all natural numbers and supports the operations of summation and multiplication. Let us denote by $\mathcal{S}$ the set of such numbers. We can extend $\mathcal{S}$ to the numbers of the form

$$a + \sqrt{2} \cdot b \qquad (*)$$

with $a, b \in \mathcal{S}$ and denote the extended numbers as $\mathcal{S}[\sqrt{2}]$. Note that depending on $\mathcal{S}$, $\mathcal{S}[\sqrt{2}]$ may be essentially the same as $\mathcal{S}$ (e.g. if $\mathcal{S}$ are all real numbers) or properly less expressive (e.g. if $\mathcal{S}$ are all rational numbers).

Implement the numbers $(*)$ in Haskell as an algebraic data type

`Sq2Num a`

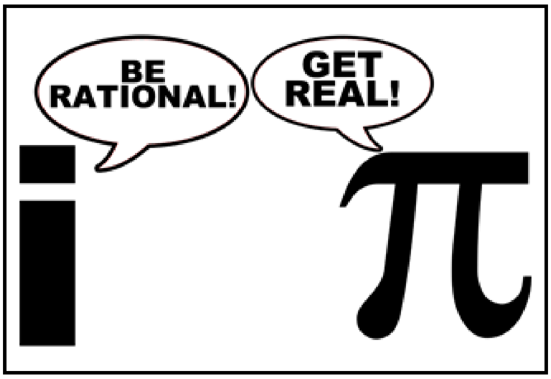
where `a` is the type capturing the elements of $\mathcal{S}$. Ensure that `Sq2Num a` (under suitable assumptions) is an instance of the following type classes: `Eq`, `Ord`, `Show`, `Num`, `Fractional`, e.g. by completing the following declarations:

```
instance (Num a, Eq a) => Eq (Sq2Num a)
instance (Num a, Eq a) => Num (Sq2Num a)
instance (Num a, Eq a, Ord a) => Ord (Sq2Num a)
instance (Fractional a, Eq a) => Fractional (Sq2Num a)
```

Additionally, provide a conversion function

```
getReal :: Floating a => Sq2Num a -> a
```

reducing from $\mathcal{S}[\sqrt{2}]$ to $\mathcal{S}$ in such a way that real numbers are converted to themselves. Like in the case of complex numbers, you need to prove (!) and implement the mathematical fact that the numbers (∗) are closed under summation and multiplication, and additionally under division, provided that so are the numbers from $\mathcal{S}$.



Hints:

- For inspiration, you can use the standard implementation of complex numbers in Haskell [1].
- That $\mathcal{S}[\sqrt{2}]$ is, for example, closed under addition follows from the fact that $\mathcal{S}$ is so, since

$$(a + \sqrt{2} \cdot b) + (a' + \sqrt{2} \cdot b') = (a + a') + \sqrt{2} \cdot (b + b')$$

You need to develop and use analogous properties for other operations.

References

[1] https://www.haskell.org/onlinereport/complex.html.