Questão 5

- ***type*** define um sinônimo de um tipo, portanto no exemplo abaixo, String e Nome referenciam exatamente o mesmo tipo, e são intercambiáveis.

```
type Nome = String;

foo :: Nome -> String;
foo x = x;

bar :: String -> Nome;
bar x = x;
```

- ***data*** define zero ou mais construtores que possuem zero ou mais valores cada. É possível definir valores com tipos que são *lazy-evaluated* (padrão) ou *strictly-evaluated* (usando exclamação). Na prática, um tipo definido com *data* precisa ser avaliado pelo seu construtor em tempo de execução, o que significa que o Haskell precisa manter o registro dos construtores durante a execução, tornando o novo tipo mais lento que o tipo original. Isso se deve à forma como o Haskell lida com *laziness*, que não permite um isomorfismo direto entre o tipo original e o tipo definido com *data*, porque eles possuem valores "*bottom*" distintos.

- ***newtype*** define exatamente um construtor que possui exatamente um valor. Um tipo A definido através de "*newtype* A = A B" funciona exatamente da mesma forma que o tipo original B, porém é necessário que um valor do tipo A seja explicitamente convertido para o tipo B, e vice-versa, caso se queira utilizar um tipo no lugar de outro. Uma vantagem do *newtype* é que, como o novo tipo é tratado pelo compilador exatamente da mesma forma que o tipo original, não há *overhead* de construtor. Tipos definidos com *newtype* são, portanto, mais eficientes que tipos definidos com *data*.

Referências

https://wiki.haskell.org/Newtype

https://www.haskell.org/onlinereport/decls.html#datatype-renaming

https://wiki.haskell.org/Bottom

https://stackoverflow.com/questions/32505911/what-is-the-role-of-bottom-%E2%8A%A5-in-haskell-function-definitions