1) Reduza a seguinte expressão lambda à forma normal:

$(\lambda x.\ \lambda y.\ (x\ (\lambda u.\ \lambda v.\ u))\ y)\ (\lambda a.\ \lambda b.\ a)\ (\lambda c.\ \lambda d.\ d)$

$(\lambda y.\ (\lambda a.\ \lambda b.\ a\ (\lambda u.\ \lambda v.\ u))\ y)\ (\lambda c.\ \lambda d.\ d)$

$(\lambda a.\ \lambda b.\ a\ (\lambda u.\ \lambda v.\ u))\ (\lambda e.\ \lambda d.\ d)$

$(\lambda b.\ (\lambda u.\ \lambda v.\ u))\ (\lambda c.\ \lambda d.\ d)$

$(\lambda u.\ \lambda v.\ u) \equiv a\ (\lambda a.\ \lambda b.\ a)$

2) compDuplas: Declare uma função que receba uma lista de duplas e retorna uma lista de duplas com os elementos das duplas invertidos.

```
compDuplas [ ] = [ ]
compDuplas [(a,b) : xs] = (b,a) : compDuplas xs
```

3) zip': Declare uma função, similar a função zip que receba como parâmetros duas listas e retorne uma lista de duplas onde o primeiro elemento da dupla pertence a primeira lista e o segundo elemento da dupla pertence a segunda lista. A execução deve ser encerrada quando uma das duplas não possuir mais elementos.

$[a] \rightarrow [b] \rightarrow [(a,b)]$

```
zip' [1,2,3,4] "abc" ⇒ [(1,'a'), (2,'b'), (3,'c')]
zip' [ ] _ = [ ]
zip' _ [ ] = [ ]
zip' (a:as) (b:bs) = (a,b) : zip' as bs
```

4) semVogal: Declare uma função que recebe uma lista de strings e retorna uma lista de strings s/ as vogais

$[String] \rightarrow [String]$

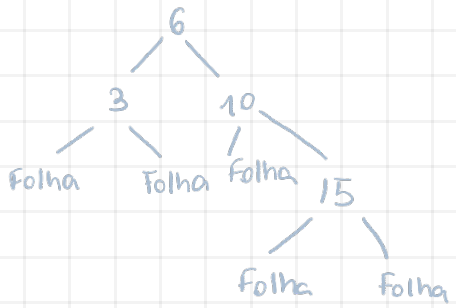
```
semvogal ["the","end"] ⇒ ["th", "nd"]
import Data.Char

semvogal [ ] = [ ]
semvogal (x : xs) = semvogal x : semvogal xs

semvogal' [ ] = [ ]
semvogal' (x: xs) | toLower x == 'a' = semvogal' xs
                  | toLower x == 'e' = semvogal' xs
                  | toLower x == 'i' = semvogal' xs
                  | toLower x == 'o' = semvogal' xs
                  | toLower x == 'u' = semvogal' xs
                  | otherwise = x : semvogal' xs
```

5) maiorArv: Dada a seguinte declaração do tipo Arvore, declare uma função que retorne o maior valor armazenado em uma árvore binária de pesquisa

```
data Arvore a = No a (Arvore a) (Arvore a) | Folha
```

Arvore a → a

```
maiorArv (No 6 (No 3 Folha Folha) (No 10 (Folha (No 15 Folha Folha)))) ⇒ 15
```




```
maiorArv' a Folha = a
maiorArv' a (No e esq dir) = maiorArv' e dir

maiorArv (No e esq dir) = maiorArv' e dir
```