Agenda Telefônica em Haskell

Disciplina: Linguagem de Programação

Docente: Carlos Bazílio Martins

Discente: Júlia Miranda Rodrigues

Contato

```haskell
data Contato = Contato {
                    nome :: [Char] -- nome do contato
                    ,telefone :: Int -- telefone do contato
                    , endereco :: [Char] -- endereço do contato
                    , relacao :: [Char] -- relação com o contato

                }deriving(Show)
```

Inserir contato

```haskell
inserir :: [Contato] -> Contato -> [Contato]
inserir agendaAntiga novoContato
                    | null agendaAntiga = [novoContato]
                    | not (null [contato | contato <- agendaAntiga, (nome contato) == (nome novoContato)]) = alterar novoContato agendaAntiga
                    | otherwise = agendaAntiga ++ [novoContato]
```

Buscar contato

```haskell
buscar :: [Char] -> [Contato] -> Contato
buscar _ [] = (Contato "Contato inexistente" 0 "-" "-")
buscar nomeDoContato (contato:agenda)
                        |estaContido nomeDoContato (nome contato) = contato
                        |otherwise = buscar nomeDoContato agenda
```

Verificação se nome está contido em outro

```haskell
estaContido_aux :: [Char] -> [Char] -> [Char]
estaContido_aux [] _ = []
estaContido_aux str1 str2
                    |(toLower( head str1) == toLower (head str2)) = [head str1] ++ (estaContido_aux (tail str1) (tail str2))
                    |otherwise = []

estaContido :: [Char] -> [Char] -> Bool
estaContido str1 [] = False
estaContido str1 str2 = if (toLower( head str1) == toLower (head str2)) && (length (tail str1) == length (estaContido_aux (tail str1) (tail str2)))
                        then True
                    else estaContido str1 (tail str2)
```

Alterar contato

```haskell
alterar :: Contato -> [Contato] -> [Contato]
alterar contatoAlterado [] = []
alterar contatoAlterado (contato : agenda)
                        |((nome contato) == (nome contatoAlterado)) = [contatoAlterado] ++ (alterar contato agenda)
                        |otherwise = [contato] ++ (alterar contatoAlterado agenda)
```

Remover contato

```haskell
remover :: [Contato] -> [Char] -> [Contato]
remover agenda nome2 = [contato | contato <- agenda, (nome contato) /= nome2]
```

Mostrar agenda

```haskell
mostrar_aux :: [Contato] -> IO()
mostrar_aux [] = putStrLn("================================================================================")
mostrar_aux (contato:agenda) = do
                               putStrLn ("Nome: " ++ nome contato
                                         ++ ", Telefone: " ++ show (telefone contato)
                                         ++ ", Endereco: " ++ endereco contato
                                         ++ ", Relacao: " ++ relacao contato)
                               mostrar_aux agenda

mostrar :: [Contato] -> IO()
mostrar agenda
                  | null agenda = putStrLn ("Agenda vazia")
                  | otherwise = do
                                putStrLn("================================================================================")
                                mostrar_aux agenda
```

Exemplo pedido

```haskell
main :: IO()
main =  do
            putStrLn("Insere Fulano na agenda vazia\n")
            let a1 = inserir [] (Contato "Fulano" 99999999 "Rua A" "UFF")
            putStrLn("\nInsere Ciclano na agenda\n")
            let a2 = inserir a1 (Contato "Ciclano" 88888888 "Rua B" "Cederj")
            putStrLn("\nInsere Beltrano na agenda\n")
            let agendaInicial = inserir a2 (Contato "Beltrano" 88889999 "Rua C" "Infância")
            putStrLn("\nInsere Fulano(novamente)\n")
            let agendaInsereFulano = inserir agendaInicial (Contato "Fulano" 77777777 "Rua D" "Churrasco do Ciclano")
            putStrLn("\nRemove Ciclano\n")
            let agendaRemoveCiclano = remover agendaInsereFulano "Ciclano"
            putStrLn("\nMostra Agenda\n")
            mostrar agendaRemoveCiclano
```

FIM