GABRIEL MOISE

FUNCTIONAL PROGRAMMING 2014

QUESTION 3

```
> unfold p f g x = if (p x) then [] else (f x) : unfold p f g (g x)
```

(a) From the if-clause, we have

```
x :: a
p :: a -> Bool
```

and the result of unfold is [b]

From the else-clause we get

```
f :: a -> b
g :: a -> a
```

Therefore, we have

```
unfold :: (a -> Bool) -> (a -> b) -> (a -> a) -> a -> [b]
```

(b)

```
> iterate :: (a -> a) -> a -> [a]
> iterate f x = x : iterate f (f x)
```

Using takeWhile and iterate, we have:

```
unfold p f g = map f . takeWhile (\ x -> p x == False) . iterate g
```

(c)

```
> data RTree a = Node a [RTree a]
> foldRTree :: (a -> [b] -> b) -> RTree a -> b
> foldRTree mode (Node a ns) = mode a (map (foldRTree mode) ns)
```

(d)

(i)

```
> dft :: RTree a -> [a]
> dft (Node a []) = [a]
> dft (Node a ns) = a : concat (map dft ns)
```

(ii)

```
> dft' :: RTree a -> [a]
> dft' = foldRTree (\ x xss -> x : concat xss)
```

(iii)

```
> unfoldRTree :: ( [RTree a] -> Bool) -> ( a -> b) -> ( [RTree a] -> [RTree a]) -> RTree a -> [b]
> unfoldRTree p f g (Node a nts) = if (p nts) then [f a] else (f a) : concat [map (unfoldRTree
p f g) (g nts))

> dft'' :: RTree a -> [a]
> dft'' = unfoldRTree null id id
```