2018 Section A

Question 2

The Code Provided:

```
data Tree = Empty
            | Single Int String
            | Many Tree Int String Tree

search :: Int -> Tree String

search x (Single i s)
    | x == i = s

search x (Many left i s right)
    | x == i = s
    | x > i = search x right
    | x < i = search x right
```

(a)

- Does not handle Empty.
- In a single, if it does not match, err
- (Also this will always go right, which is not good)

(b) The solution with maybe:

```
data Tree = Empty
    | Single Int String
    | Many Tree Int String Tree

search :: Int -> Tree -> Maybe String

search _ Empty = Nothing

search x (Single i s)
    | x == i = Just s
    | otherwise = Nothing

search x (Many left i s right)
    | x == i = Just s
    | x > i = search x right
    | x < i = search x left
```

(c) The solution with monads:

```haskell
data Tree = Empty
    | Single Int String
    | Many Tree Int String Tree

search :: Monad m => Int -> Tree -> m String

search _ Empty = fail "Not in tree!!"

search x (Single i s)
    | x == i = return s
    | otherwise = fail "Not in tree!!"

search x (Many left i s right)
    | x == i = return s
    | x > i = search x right
    | x < i = search x left
```