EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät

Programmiersprachen

Prof. Klaus Ostermann

Steven Lolong
steven.lolong@uni-tuebingen.de

Object Meets Function

Assignment 5 – Winter Semester 23/24

Tübingen, 6. Februar 2024

Handin Please submit the homework until Sunday, Feb. 19th 2023 via email to Steven Lolong (steven.lolong@uni-tuebingen.de) before 24:00.

Mandatory This assignment is a mandatory, every participant should submit the solution of the assignment.

Email Format Use this format for email's subject: **OmF-W22/23-Assign**[no]**-**[YourName]

File name Format Use this format for file name **Assig-**[no]-[Your Name]

Task 1: Parametric Type on Binary Tree (12 Points)

Before you work on Task 1, you should design ADT for the binary tree below.

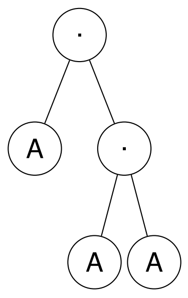

Abbildung 1: **A = Parametric type**

1.1 Operation on Binary Tree (6 Points)

Regarding the tree above, implementing operations for every type requires a lot of work. So, to avoid too much work for every type, please write the map function, filter function, and fold function for your binary tree.

1. The map function (2 points).
2. The filter function, for this function, rather than create a new tree structure, you only need to delete the content of the leaf with the id of the type (where the id is a part of the function parameter). (2 points)
3. The fold function (2 points)

1.2 Proof of Functions (6 Points)

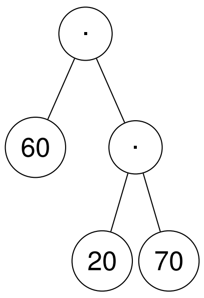

Abbildung 2: **BT-Integer**

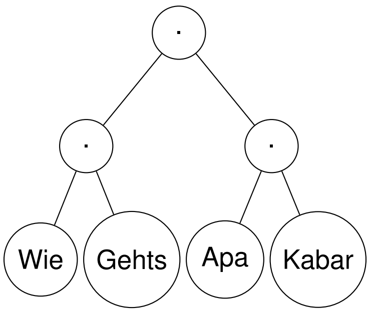

Abbildung 3: **BT-String**

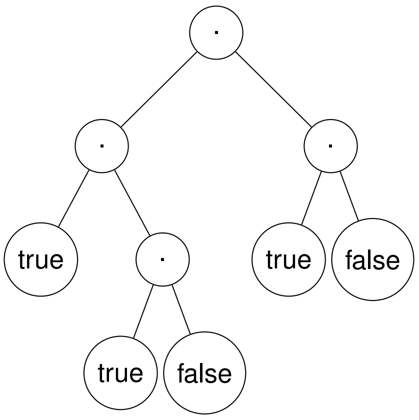

Abbildung 4: **BT-Boolean**

1. Please sum abbildung 2: BT-Integer using fold function, the result is150. (2 points)
2. Using filter function, please choose only 'Wie' (abbildung 3: BT-String) will stay in the leaf other than that will replace by id (id of string is empty string). (2 points)
3. Please change all false to true (see abbildung 4: BT-Boolean). Rather than check and replace, please using map function to apply on it with or (partial or with true, ex: _ || $true$). (2 points)

Task 2: Monad Transformation (16 Points)

2.1 Please complete the declaration of MonadTransformation below on line 27 and 28. (8 points)

```
 1 trait Monad[M[_]]:
 2   def unit[A](a: A): M[A]
 3   def bind[A, B](m: M[A], f: A => M[B]): M[B]
 4 end Monad
 5
 6 // List monad
 7 object ListMonad extends Monad[List]:
 8   override def unit[A](a: A): List[A] = List(a)
 9   override def bind[A, B](m: List[A], f: A => List[B]): List[B] = m.flatMap(
     f)
10 end ListMonad
11
12 // create a type mapping (transformation)
13 type OptionT[M[_]] = [A] =>> M[Option[A]]
14
15 // take Monad List and transform into Monad Option
16 class MonadTransformation[M[_]](mlist: Monad[M]) extends Monad[OptionT[M]]:
17   override def unit[A](a: A): OptionT[M][A] = mlist.unit(Some(a))
18
19   override def bind[A, B](
20       m: OptionT[M][A],
21       f: A => OptionT[M][B]
22   ): OptionT[M][B] =
23     mlist.bind(
      m,
      (z: Option[A]) =>
        z match
          case Some(v) => __________________
          case None    => __________________
    )
end MonadTransformation
```

Listing 1: Monad

2.2 Please complete the test case below to prove your MonadTransformation is correct. (8 points)

```
    val mt = new MonadTransformation(_________)

    def mt_case1 = mt.bind(_______, ________)
    // expected output when call mt_case1 is List(Some(3), Some(5))

    def mt_case2 = mt.bind(_______, ________)
    // expected output when call mt_case2 is OptionT[List][Int] = List(Some
   (4), None, Some(5))

    def mt_case3 = mt.bind(_____________, _______________)
    // expected output when call mt_case3 is OptionT[List][String] = List(
   Some(3), None, Some(4))
```