Appendix of
"Mind the Error Message"

Kazuhiro Tsunoda
k-tsunoda@prg.is.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

Hidehiko Masuhara
masuhara@acm.org
Tokyo Institute of Technology
Tokyo, Japan

Youyou Cong
cong@c.titech.ac.jp
Tokyo Institute of Technology
Tokyo, Japan

ABSTRACT

This is appendix of the paper "Mind the Error Message: an Inverted Quiz Format to Direct Learner's Attention to Error Messages"

ACM Reference Format:
Kazuhiro Tsunoda, Hidehiko Masuhara, and Youyou Cong. 2023. Appendix of "Mind the Error Message". In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023), July 8–12, 2023, Turku, Finland*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3587102.3588823

A TASKS AND RESULTS OF PRELIMINARY EXPERIMENT

Overview. Before the preliminary experiment, we asked three participants about their programming experience in the questionnaire (Table 1).

In the preliminary experiment, we performed two debugging-style exercises. We showed the two problems in Listing 1 and 2 in order through screen sharing, and we asked participants to verbally answer which part of the source code should be changed to solve the error. For the first question, we gave the participants 3 minutes to answer. If the participant failed to answer, we gave an explanation of the correct solution. For the second question, we hid the problem after 25 seconds, regardless of whether or not the student had answered the question. Then we asked them to answer what was written in the error message to the extent that they remembered.

Results. In the first question, all students could come up with an appropriate solution to solve the error within 3 minutes. In the second question, two students who had less programming experience answered "It said that the parameter `m` was undefined/reassigned."" This suggests that they payed attention to the non-essential part of the error message. On the other hand, one student with a longer programming experience answered "It said that there were not enough arguments for `Child`", which was the essential part of the error message.

Listing 1: Question 1

```
def xor(a: Boolean, b: Boolean): Boolean = {
  if (a = !b) true
  else false
}

//error message
// .../testq1.scala ;2;6; reassignment to val
// if(a = !b) true
//      ^
```

Listing 2: Question 2

```
abstract class FTree
case class End(name: String) extends FTree
case class Child(n: String, f: FTree, m: FTree) extends
     FTree

def attach(tree: FTree): FTree = {
  tree match {
    case End(name) => End(name + "san")
    case Child(n,f,m) => Child(attach(f),attach(m))
  }
}

//error message
// .../testq2.scala:8:31: not enough arguments for method
     apply: (n: String, f: FTree, m: FTree)Child in object
      Child.
// Unspecified value parameter m.
// case Child(n,f,m) => Child(attach(f),attach(m))
//                              ^
```

Table 1: Result of preliminary experiment's questions

	student1	student2	student3
programming experience (Year)	〜1	1〜2	3〜5
question1	success	success	success
question2	inaccurate	inaccurate	accurate

ITiCSE 2023, July 8–12, 2023, Turku, Finland
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0138-2/23/07...$15.00
https://doi.org/10.1145/3587102.3588823

B ENBUGGING QUIZZES USED IN MAIN EXPERIMENT

Listing 3: day5

```
//question code
def append(x: Int, list: List[Int]): List[Int] = {
  x:: list
}

//expected error message
// Found: (true : Boolean)
// Required: Int
```

Listing 4: day6

```
//question code
case class Color(red: Int,green: Int,blue: Int)

def revColor(c: Color): Color = {
  c match{
    case Color(r,g,b) => Color(255-r,255-g,255-b)
    }
}

//expected error message
// missing argument for parameter blue of method apply in object Color: (red: Int, green: Int, blue: Int): Playground.Color
```

Listing 5: day7

```
//question code
def head(l1: List[Int]): Int = {
  l1 match{
    case Nil => 0
    case x::xs => x
  }
}

//expected error message
// Unreachable case
```

Listing 6: day8

```
//question code
def makepairs(l1: List[Int],l2: List[Int]) : List[(Int,Int)] = {
  (l1,l2) match {
    case (Nil,_) => Nil
    case (_,Nil) => Nil
    case (x1::x1s,x2::x2s) => (x1,x2)::makepairs(x1s,x2s)
  }
}

//expected error message
// match may not be exhaustive.
//
// It would fail on pattern case: (Nil, List(_, _*))
```

Listing 7: day9

```
//question code
def ifprint(x:Int)(y:Boolean) : Int = {
  if(y) print(x)
  x
}

def print(x:Int): Int = {
  ifprint(x)(true)
}

//expected error message
// Found: Boolean => Int
// Required: Int
```

Listing 8: day10

```
//question code
def grater[A](x: Int,y: Int): Boolean = {
  x > y
}

//expected error message
// None of the overloaded alternatives of method > in class Int with types
// (x: Double): Boolean
// (x: Float): Boolean
// (x: Long): Boolean
// (x: Int): Boolean
// (x: Char): Boolean
// (x: Short): Boolean
// (x: Byte): Boolean
// match arguments ((y : A))
```

Listing 9: day11

```
//question code
def not(x: Boolean): Boolean = {
  if (x == true) false
  else true
}

//expected error message
// A pure expression does nothing in statement position; you may be omitting necessary parentheses
```

Listing 10: day12

```
//question code
def numbering(list: List[String]): List[(String, Int)] = {
    def subnum(list: List[String], pos: Int): List[(String, Int)] = {
        list match {
          case Nil => Nil
          case x :: xs => (x, pos) :: subnum(xs, pos + 1)
        }
    }
  subnum(list, 0)
}

//expected error message
// Found: Unit
// Required: List[(String, Int)]
```

C ABOUT CORRECTION

We found an incorrect quiz (day12) posted, so it was corrected on October 28, 2023. The incorrect code was below. Also, we found that the number of boxes on day9 and day11 was in the old version, so revised it to the number that was actually used one.

Listing 11: incorrect day12

```
//question code
def plus_minus(list: List[Int]) : Int = {
  def sub_p_m(list: List[Int],p_m:Int) : Int = {
    list match{
      case Nil => 0
      case x::xs => x * p_m + sub_p_m(xs,p_m*(-1))
      }
    }
  sub_p_m(list,1)
}

//expected error message
// Found: Unit
// Required: List[(String, Int)]
```