CSPB 3155 - Reckwerdt - Principles of Programming Languages

Started on	Thursday, 27 June 2024, 2:28 PM
State	Finished
Completed on	Thursday, 27 June 2024, 2:41 PM
Time taken	13 mins 12 secs
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 2.00 out of 2.00

Consider the following scala code with case pattern matching.

```
sealed trait A
case class X ( i : Int ) extends A
case class Y ( j : Int , x : A ) extends A
case object Z extends A

def unwind ( a : A ) = a match {

   case X ( j ) =>  { .. pattern number 1 .. }
   case Z  => {... pattern number 2 .. }
   case Y ( j , Z )  =>  {... pattern number 3 .. }
   case Y ( j , X ( i )) if i < 0 =>  { .. pattern number 4 ..}
   case X ( j ) =>  {.. pattern number 5 .. }

}
```

(A, 2 points) Consider the call: `unwind( Y( 20, X(-15) ) )`

- Which case pattern number (1..5) will it match? 4 ✔
- What value will the identifier j bind to? 20 ✔

(B: 2 points): Which of the following patterns would not successfully match if used as arguments to unwind? Select one:

- (•) Y(20,X(35)) ✔
- ( ) X(25)
- ( ) Z
- ( ) Y(100,X(-50))

Mark 2.00 out of 2.00

The correct answer is: Y(20,X(35))

Question **2**

Correct

Mark 2.00 out of 2.00

Write down the value that the following Lettuce expression will yield under the environments provided? If the value is a number, simply write the number down (do **not** write NumValue(number)). If it is boolean, simply write true/false as case may be (do not write Boolvalue(boolean)). Finally, if an **error** results, just write **error (small letters).**

(A: 2 points) (5 + z) in environment [z -> 15, x -> 20] 20 ✔

(B: 2 points) (y + z) in environment [z -> 15] error ✔

(C: 2 points) (y + z) in environment [z -> 15 , y -> 5] 20 ✔

(D: 2 points) (y + z) in environment [z -> 15, y -> true, x -> 20] error ✔

Question **3**

Correct

Mark 6.00 out of 6.00

Some languages such as C/C++ allow if-then-else statements whose conditions can be numbers.

If **the number is non-zero**, the **condition is taken to be true** and **if it equals zero**, the **condition is taken to be false**.

(A, 2 points) Under these semantics, what is the value computed by the following expression:

```
if (10)
then 5
else 4
```

The value of the expression is 5 ✔

(B, 7 points) Let us write semantics rules to evaluate if-then-else statements in Lettuce according to the new semantics. In the rule we have placeholders such as Q1, Q2... and so on, that need to be selected from the options provided.

Write a rule that says that

Evaluating the expression **IfThenElse(e, e1, e2)** under env is the same as evaluating e1 under env if e evaluates to a non-zero number under env

$$\frac{eval(e,env) = v,\ v \in R,\quad Q1,}{eval(IfThenElse(e,e1,e2),env) = Q2}\ (if-then-else-non-zero)$$

Q1: v ≠ 0 ✔ (2 points)

Q2: eval(e1, env) ✔ (1 point)

Write a rule for evaluating *IfThenElse(e, e1, e2)* under env. env, if e evaluates to 0.

Evaluating the expression **IfThenElse(e, e1, e2)** under env is the same as evaluating e2 under env if e evaluates to the number zero under env.

$$\frac{eval(e,env) = v,\ v \in R,\quad Q3,}{eval(IfThenElse(e,e1,e2),\sigma) = Q4}\ (if-then-else-zero)$$

Q3: v = 0 ✔ (2 points)

Q4: eval(e2, env) ✔ (2 points)

(C, 6 points) We try to modify the original code for the interpreter in Scala. The relevant portion for handling IfThenElse is shown bellow. Complete the code stubs:

```
def eval ( e : Expr , sigma : Map [ String , Expr ]): Value = e match {
    .... //Other Stuff Skipped
    case IfThenElse (e , e1 , e2 ) = > {
    val v = eval(e , sigma )
    v match {
       case True = > { .... }
```

```
       case False => { .... }
```

```
      case Const( n ) = > {
           if ( n != 0 ) { eval(e1,sigma) ✔ } // if-then-else-non-zero (3 points)
           else { eval(e2,sigma) ✔ } // if-then-else-zero (3 points)
       }
      case _ = > { throw new IllegalArgumentException ( "boohoo")}
   }
   ... //Other Stuff Skipped
}
```