CSPB 3155 - Reckwerdt - Principles of Programming Languages

Dashboard / My courses / 2244:CSPB 3155 / Week 3: Inductive Definitions and Case Pattern Matching / Online Quiz 3

Started on	Saturday, 1 June 2024, 9:00 PM
State	Finished
Completed on	Saturday, 1 June 2024, 9:04 PM
Time taken	3 mins 35 secs
Marks	15.00/15.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 3.00 out of 3.00

For this quiz you will need to go over the Mython grammar defined in the notebook on "Inductive Definitions"

(Also found in this week's moodle under 'Resources')

We wish to add a statement to the mython language that will print the value of an arithmetic expression to the screen

```
print(expr)
```

Select the appropriate grammar rule to be added that achieves this.

Select one:

○ a. **Statement ->** Print(**CondExpr)**

○ b. **Expr ->** Print(**Expr)**

○ c. **Program ->** Print(**Expr)**

● d. **Statement ->** Print(**Expr)** Correct

○ e. **PrintStatement ->** Print(**Expr)**

Your answer is correct.

The correct answer is: **Statement ->** Print(**Expr)**

Correct

Marks for this submission: 3.00/3.00.

Question **2**

Correct

Mark 3.00 out of 3.00

We wish to add a python style for loop to mython

```
for (i from expr to expr) {
 // statements in the body of the loop
}
```

Which of the grammar rules below is appropriate for adding a for loop?

Select one:

○ a. **Statement ->** For(**Identifier, CondExpr, CondExpr, Statement*)**

○ b. **Statement ->** For(**Identifier, From, Expr, To, Expr, Statement*)**

○ c. **Statement ->** For(**Expr, Expr, Statement*)**

● d. **Statement ->** For(**Identifier, Expr, Expr, Statement*)** Correct

○ e. **Statement ->** For(**Identifier, Expr, Statement*)**

Your answer is correct.

The correct answer is: **Statement ->** For(**Identifier, Expr, Expr, Statement*)**

Correct

Marks for this submission: 3.00/3.00.

Question **3**

Correct

Mark 3.00 out of 3.00

Consider the definition of a `NumTree`

```
sealed trait NumTree
case class Leaf(k: Int) extends NumTree
case class Node(k: Int, lChild: NumTree, rChild: NumTree) extends NumTree
```

let `v` be a `NumTree` value and suppose we wrote a pattern matching statement:

```
 v match {
       case Leaf(k) if k >= 0 =>  // Case number 1
       case Leaf(k) if k <= 0 => // Case number 2
       case Leaf(k) if k == 0 => // Case number 3
       case  Node(k, Leaf(_) , rChild) if k >= 0 => // Case number 4
       case Node(k, _, _) => // Case Number 5
}
```

Select which case each of the possible values will match. Refer to cases by the provided numbers.

An `if` which comes before the => is called a "pattern guard".

Leaf(10)	Case number 1
Leaf(0)	Case number 1
Node(10, Leaf(20), Node(5, Leaf(10), Leaf(10)))	Case number 4
Node(10, Node(5, Leaf(10), Leaf(10)), Leaf(20))	Case number 5
Node(-5, Leaf(-5), Node(5, Leaf(10), Leaf(10)))	Case number 5

Your answer is correct.

The correct answer is: Leaf(10) → Case number 1, Leaf(0) → Case number 1, Node(10, Leaf(20), Node(5, Leaf(10), Leaf(10))) → Case number 4, Node(10, Node(5, Leaf(10), Leaf(10)), Leaf(20)) → Case number 5, Node(-5, Leaf(-5), Node(5, Leaf(10), Leaf(10))) → Case number 5

Correct

Marks for this submission: 3.00/3.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Consider the inductive definition of a list

```
sealed trait NumList
case object Nil extends NumList
case class Cons(hd: Int, tl: NumList) extends Numlist
```

Let

```
l
```

be a value of the type `NumList`. Which of the following pattern match statements on `l` are exhaustive: I.e, every possible value of `l` matches with some case?

Select one or more:

☑ a.

```
l match {
     case Nil => ...
     case Cons(a, _ ) => ...
 }
```

Exhaustive since a list is either a Nil or Cons(some integer, rest)

☑ b.

```
l match {
     case Nil => ...
     case Cons(a, Nil) => ...
     case _ => ...
 }
```

Exhaustive because we have case _ that will match anything.

□ c.

```
l match {
     case Cons(a, Cons(b, Nil) )  => ..
     case Cons(a, Nil) =>
}
```

□ d.

```
l match {
     case Nil => ...
     case Cons(a, Cons(b, Nil) )  => ..
     case Cons(a, Nil) =>
}
```

□ e.

```
l match {
     case Nil => ...
     case Cons(a, k ) if a >= 0  => ...
 }
```

Your answer is correct.

The correct answers are:

```
l match {
     case Nil => ...
     case Cons(a, Nil) => ...
     case _ => ...
 }
```

,

```
l match {
    case Nil => ...
    case Cons(a, _ ) => ...
 }
```

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 5.00 out of 5.00

Consider the definition:

```
sealed trait NumTree
case class Leaf(k: Int) extends NumTree
case class Node(k: Int, n1: NumTree, n2: NumTree) extends NumTree
```

And the pattern matching for a value v of type `NumTree`

```
  v match {
       case Leaf(k) => ... // Case number 1
       case Node(k, Leaf(k1), rChild) if k1 >= k => ... // Case number 2
       case Node(k, nd1@Node(k1, _, _), nd2@Node(k2, lChild2, rChild2) ) if k == k1 && k <= k2 => .. // case 3
       case _ => // case 4
  }
```

Consider the value below which matches case number 2:

```
Node(10, Leaf(20), Leaf(30))
```

What is the value bound to `rChild`: enter your answer without whitespaces? Leaf(30)

What is the value bound to k1: enter your answer without whitespaces? 20

Consider the value that matches case number 3.

```
Node(10, Node(10, Leaf(5), Leaf(5)) , Node(30, Leaf(4), Leaf(8)) )
```

What is the value bound to `nd1`? Make sure that there are no white spaces in your answer

Node(10, Leaf(5), Leaf(5))

What is the value bound to `rChild2`? Leaf(8)

What is the value bound to k2? 30

Correct

Marks for this submission: 5.00/5.00.