Quiz 4 (Semantics & Types) Results for Ya Zou

ⓘ Answers will be shown after your last attempt

Score for this attempt: **100** out of 100

Submitted Feb 26 at 10:43am

This attempt took 46 minutes.

Question 1

10 / 10 pts

Consider the following data type that represents the abstract syntax of some unknown language.

```
data S = A Int
       | B Int Int S
```

Which of the following types corresponds most closely to S and could thus be used as an alternative abstract syntax?

○ (Int, [Int])

○ [(Int, Int)]

● (Int, [(Int,Int)])

○ [Int]

○ None of these

○ ([Int], [Int])

Question 2

10 / 10 pts

Consider the following abstract syntax for a simple expression language. The Plus operation results in an integer, the result of Equal operation is a boolean and the Not operator can only be applied to boolean types.

```
data Exp = Con Int | Plus Exp Exp | Equal Exp Exp| Not Expr
```

What is a proper semantic domain for defining the denotational semantics?

○ Int

○ Either Int Bool

○ Either (Maybe Int)(Maybe Bool)

● Maybe (Either Int Bool)

○ Maybe Int

Question 3

10 / 10 pts

Consider the follow syntax excerpt from a language for computing with number and lists of numbers.

exp ::= ... | *num* | [] | *exp*:*exp* | head *exp*

Which of the following type definitions for D are appropriate semantic domains for defining the denotational semantics of the language? You can select more than one.

☑ data D = N Int | List [Int] | Error

data Val = N Int | List [Int]

☑ type D = Maybe Val

□ type D = (Int, [Int])

□ data D = N Int | [Int]

Question 4

10 / 10 pts

Consider the following abstract syntax for a language for non-nested integer lists. N represents integer constants. The constant Empty denotes an empty list. The operation Cons adds an integer (given as the first argument) to a list. We can extract the first element of a list using Head and the operation Length represents a function to compute the length of a list.

```
data Expr = N Int | Empty | Cons Expr Expr | Head Expr | Length Expr
```

Which of the following expressions should be considered to be type correct by a type checker for that language? Select one or more.

☑ Cons (Length Empty) Empty

☑ Length Empty

□ Cons (N 1) (N 5)

☑ Head (Cons (N 5) Empty)

□ Cons (Length Empty)

Question 5

10 / 10 pts

Complete the semantics code for a simple expression language with two types by selecting the code that replaces the ??????

```
data Val = I Int
         | B Bool
         | Err

sem :: Expr -> Val
sem (N i)        = I i
sem (Plus e e')  = case (sem e,sem e') of
                     (I i,I j) -> I (i+j)
                     _         -> Err
sem (Equal e e') = case (sem e,sem e') of
                     (I i,I j) -> B (i==j)
                     (B b,B c) -> ????????
                     _         -> Err
```

○ None of these

○ B b == B c

○ b == c

● B (b==c)

○ b && c

Question 6

10 / 10 pts

Complete the semantics code for a Boolean expression language with only Boolean types

```
data BExpr = T | F
           | Not BExpr
           | Or BExpr BExpr
           | And BExpr BExpr

sem :: BExpr -> Bool
sem T         = True
sem F         = False
sem (Not b)   = not (sem b)
sem (Or b b') = sem b || sem b'
```

Select the code to add the "And" operation to sem.

○ sem (And b b') = True

○ sem (b && b')

○ sem (b And b') = b && b'

● sem (And b b') = sem b && sem b'

Question 7

10 / 10 pts

Select **ALL** examples of the type [(Int, Bool)]

☑ [(5, True), (6, False)]

☐ [(5, T), (6, F)]

☐ (5, True)

☑ []

☐ [( )]

Question 8

10 / 10 pts

Select **ALL** examples of the type Maybe [Bool]

☑ Just [True, False, True]

☐ [Nothing]

☐ [Just True, Just False]

☑ Nothing

☐ Just True

Question 9

10 / 10 pts

Conider the type checker for a simple expression language

```
data Type = Int | Bool | TypeError
            deriving (Eq,Show)

tc :: Expr -> Type
tc (N i)                                    = Int
tc (Plus e e')  | tc e==Int  && tc e'==Int  = Int
tc (Equal e e') | tc e==Int  && tc e'==Int  = Bool
                | tc e==Bool && tc e'==Bool = Bool
tc (Not e)      | tc e==Bool                = Bool
tc _                                        = TypeError
```

Suppose you want to add type checking for integer multiplication (Mult expr expr) of two expressions that must evaluate to integers. Select the appropriate line of code.

○ tc (Mult e e') = (e * e') == Int

○ tc (Mult e e') = Int e * Int e'

● tc (Mult e e') | tc e == Int && tc e'==Int = Int

○ tc (Mult Int Int) = Int

Question 10

10 / 10 pts

What types are determined for the following expression under static and dynamic typing?

Always assume strong typing, and make an optimistic assumption about the type of the variable x, that is, assume a type for x that makes the expression as type correct as possible.

```
if x < 5 then even x else x
```

○ Static: Bool
Dynamic: Type Error

○ Static: Type Error
Dynamic: Int

○ Static: Type Error
Dynamic Type Error

● Static : Type Error
Dynamic: Bool if x < 5, otherwise Int

Quiz Score: **100** out of 100