Introduction to Programming: Quiz 2

Date: October 26, 2026 **Time:** 1145 – 1215 hrs **Marks:** 25 **Weightage:** 10%

Part A has eight multiple-choice questions of 2 marks each. Circle (or tick) exactly one option.
Part B has one program-tracing question worth 9. Show your working.
Use the lecture definitions unless stated otherwise.
Use your own paper (or page 4) for rough work. Clearly mark the rough work portion.
Write your name and roll number on top.

Part A: Multiple choice

(8 × 2 = 16 marks)

Questions 1–3 use these declarations:

```
data Day = Sun | Mon | Tue | Wed | Thu | Fri | Sat
  deriving (Eq, Show, Ord, Enum)

data Shape = Circle Double | Square Double | Rectangle Double Double
  deriving (Eq, Ord, Show)
```

1. What does `[d | d ← [Sun ..], d > Thu]` evaluate to?

   (A) `[Fri,Sat]` (B) `[Thu,Fri,Sat]` (C) `[Sun,Mon,Tue,Wed]`

   (D) Compile-time error: `Day` is not a numeric type, so `[Sun ..]` is not allowed

2. Which of the following evaluates to **`True`**?

   (A) `Square 1.0 < Circle 10.0`

   (B) `Rectangle 1.0 5.0 < Rectangle 1.0 2.0`

   (C) `Circle 9.0 < Square 1.0` (D) `Square 2.0 > Rectangle 1.0 1.0`

3. What is the type of `map Rectangle [1.0, 2.0]`?

   (A) `[Shape]` (B) `[Double → Shape]` (C) `Double → [Shape]`

   (D) Compile-time error: `Rectangle` needs two arguments

4. What is the most general type of `f`?

```
f x ys = length (filter (== x) ys) > 0
```

   (A) `a → [a] → Bool` (B) `Ord a ⇒ a → [a] → Bool`

   (C) `(Eq a, Num a) ⇒ a → [a] → Bool`

   (D) `Eq a ⇒ a → [a] → Bool`

5. Consider the following code.

```
data Coin = H | T deriving (Eq, Show)

instance Show Coin where
  show H = "Heads"
  show T = "Tails"
```

What happens when we load this file and evaluate `show [H, T]`?

(A) `"[Heads,Tails]"` (B) `"[H,T]"` (C) `"HeadsTails"`

(D) Compile-time error: duplicate **`Show`** `Coin` instance

6. Which expression is rejected by the type checker?

(A) `elem (1, 'a') [(1, 'a'), (2, 'b')]`

(B) `max "abc" "abd"` (C) `[(+3)] == [(*5)]`

(D) `read "42" + (1 :: Int)`

7. What happens when this definition is compiled?

```
average :: [Double] → Double
average xs = sum xs / length xs
```

(A) Compile-time error: `length xs` has type **`Int`**, but `/` needs both operands to have type **`Double`**

(B) It compiles, and `average [1, 2, 3]` returns `2.0`

(C) Compile-time error: **`Double`** is not an instance of **`Num`**

(D) It compiles, but `average []` crashes with a divide-by-zero exception

8. Each expression is typed at the `ghci` prompt. Which one *never finishes* printing its result?

(A) `length (map (`div` 0) [0..9])` (B) `take 3 (filter even [1..])`

(C) `head (2 : reverse [1..])` (D) `filter (< 5) [1..]`

Part B: Program tracing (9 marks)

9. Consider the following code.

```
pick :: Int → [Int] → Int
pick _ []     = 0
pick 0 (x:_)  = x
pick n (_:xs) = pick (n-1) xs

nums :: [Int]
nums = map (*3) [1 ..]
```

[5] (a) Show the leftmost-outermost reduction of `pick (4-2) nums` until no further simplification is possible. You may combine steps that unfold `map` and `[1 ..]` into a single step.

[4] (b) What do the following return? Justify each answer in one line.

(i) `pick 1 [5 `div` 0, 7, 8]`

(ii) `pick 1 [5, 7 `div` 0, 8]`