Type-Directed Programming Quiz

1. Type-directed programming is a language mechanism that *infers values from types*.
2. When the compiler infers an implicit parameter of type T, it searches for possible candidates in:
    - *Imported implicit definitions*
    - *Implicit definitions in companion objects of the types associated with the type T*
    - *Inherited implicit definitions*
    - *Implicit definitions of outer scopes*
3. What is the output of the following program?

```
implicit val n: Int = 42
def f(implicit x: Int) = x
println(f)
```

   Answer: *42*
4. What is the output of the following program?

```
implicit val n: Int = 42
def f(implicit x: Int) = x
println(f(0))
```

   Answer: *0*
5. How could you change the first line of this program to make it compile?

```
val world: String = "World"
def greet(implicit name: String) = s"Hello, $name!"
println(greet)
```

   Answer: *implicit val world: String = "World"*
6. What is the output of the following program?

```
trait LowPriorityImplicits {
    implicit val intOrdering: Ordering[Int] = Ordering.Int
}
object Main extends LowPriorityImplicits {
    implicit val intReverseOrdering: Ordering[Int] = Ordering.Int.reverse
    def main(args: Array[String]): Unit = {
        println(List(1, 2, 3).min)
    }
}
```

   Answer: *3*
7. Consider the following program:

```
trait Show[A] {
    def apply(a: A): String
}
object Show {
    implicit val showInt: Show[Int] = new Show[Int] {
        def apply(n: Int): String = s"Int($n)"
    }
}
implicitly[Show[Int]]
```

Take a close look at the last line. The expression implicitly[Show[Int]] compiles because the compiler finds an implicit value of type Show[Int] and supplies it as an argument to the method implicitly. For reference, here is the definition of implicitly:

```
def  implicitly[A](implicit arg: A): A = arg
```

Rewrite the last line to show explicitly the implicit argument that has been inferred by the complier.

Answer: *implicitly[Show[Int]](Show.showInt)*

8. What is the output of the following program?

```
trait Show[A] {
    def apply(a: A): String
}
object Show {
    implicit val showInt: Show[Int] = new Show[Int] {
        def apply(n: Int): String = s"Int($n)"
    }
}
def printValue[A: Show](a: A): Unit = {
    println(implicitly[Show[A]].apply(a))
}
printValue(42)
```

Answer: *Int(42)*