CSCI 3155 – Final Exam

This exam is due Sunday, May 3rd by 23:55. You should submit your exam to Moodle. Keep your answers short and concise. Use your own words and examples – do not simply copy from the text or Internet. What you turn into me must by typed, not hand-written, save for those occasions when a drawing or formula would take too much time to attempt in Word or other word-processing program.

1. Some programming languages are typeless. What are the obvious advantages and disadvantages of having no types in a language?
2. What are the arguments for and against representing Boolean values as single bits in memory?
3. How does a decimal value waste memory space?
4. Dynamic type binding is closely related to implicit heap-dynamic variables. Explain this relationship.
5. What are the primitive data types in Scala?
6. You are given the following Scala code:

   ```
   class C(x: Int) {}
   class D(x: Int) {}
   c = d
   ```

   What is the problem with the code?

7. You are given the following Scala code:

   ```
   val numbers = List(1, 2, 3, 4)
   numbers.map((i: Int) => i * 2)
   numbers.foreach((i: Int) => i * 2)
   ```

   Why do the map and foreach calls produce different results?

8. You are given the following Scala code:

   ```
   import scala.language.reflectiveCalls
   def foo(x: { def get: Int }) = 123 + x.get
   foo(new { def get = 10 })
   ```

   What is the output of these commands?

   What kind of **type compatibility** is implicit in this?

9. You are given the following Scala code:

```
      class someDS[A] {
              private class Node[A] (elem: A) {
              var next: Node[A] = _
              override def toString = elem.toString
          }
      private var head: Node[A] = _

      def add(elem: A) {
      val n = new Node(elem)
      n.next = head
          head = n
  }

  private def printNodes(n: Node[A]) {
          if (n != null) {
          println(n)
          printNodes(n.next)
          }
  }

  def printAll() { printNodes(head) }

  }
```

a) What type of data structure does this code implement?
b) What data type(s) does this class accept?
c) Give an example of how you would use it.

10. What type of operator overloading does Scala permit?

11. In Scala operators are actually methods. Provide an example of a method using traditional dot notation and its equivalent as an operator.

12. In Scala, match is used instead of switch. Why is this advantageous in comparison to how switch is implemented in C/C++? Provide at least two reasons.

13. Why does the following code fail in Scala and how would you fix it?

```
val list = 1::2::3
```

14. Scala does not include the keywords **break** and **continue**.

    a. Why not?
    b. How does Scala provide equivalent functionality?

15. Explain why it is difficult to eliminate functional side effects in C.

16. Scala provides closure functionality.

    a. Define what a closure is and provide a Scala example.
    b. What conditions are necessary for a language to allow for closures and why is this so?

17. Briefly describe the three methods used to provide synchronization in the context of concurrency.

18. What types of synchronization do Java and Scala provide? What are the advantages to Scala's approach?

19. Scala uses Actors to implement concurrency. (Note that the latest versions of Scala use the Akka actor library but the set-up is a bit more involved so I stuck with the Actors library because it is covered in the 2nd edition of the Odersky book.) Below is a simple version of a program that creates two actors and runs them concurrently.

```
package actors

import scala.actors._

object SillyActor extends Actor {
  def act() {
    for (i <- 1 to 5) {
      println("I'm acting!")
      Thread.sleep(1000)
    }
  }
}

object SeriousActor extends Actor {
  def act() {
    for (i <- 1 to 5) {
      println("To be or not to be.")
      Thread.sleep(1000)
    }
  }
}

object Actors {
  def main(args: Array[String]) {
    SillyActor.start()
    SeriousActor.start()
  }
}
```

    a. Run this example and list its output.
    b. Comment the code explaining what is happening.
    c. Add a third actor with a longer sleep time and run the code listing the output.