CS245 Final Exam

Name:

Start Time:

Finish Time:

<u>You must finish and submit within 200 minutes of downloading this test</u>

If you have given me a testing accommodation letter please indicate that here by briefly stating the terms of you accommodation. (eg., "quiet room")

I have abided by the Honor Code. I have not discussed, nor will I discuss, this test with anyone.
(Sign below)

If you take this test on separate sheets of paper, make the above your first page.

There are 11 questions in this test. Every question is worth 9 points. (There is one point for filling out this page.) Be sure to answer all of the questions.

1. Describe the difference between scope and extent. Can scope be different from extent in a PL with third class functions? Explain.

2. Iterators are useful and Go has them for slices and and maps. However, the Linked List below does not have a usable iterator. So write one. The iterator should be usable as illustrated in the main function.

```
package main

import "fmt"

type Node struct {
    next *Node
    data int
}

func (n *Node) add(nnode *Node) bool {
    if n.next!=nil {
        return false
    } else {
        n.next = nnode
        return true
    }
}

func main() {
    n1:=Node{data:1}
    n2:=Node{data:2}
    n3:=Node{data:3}
    n1.add(&n2)
    n2.add(&n3)
    n3.add(&Node{nil,4})
    n3.next.add(&Node{data:5})
    for getNext:=n1.getIterator();; {
        v:=getNext()
        if v==nil {
            break;
        }
        fmt.Printf("%v\n", v.data)
    }
}
```

3. The Kotlin function mapIndexNotNull might be used as follows:

```
fun main(args: Array<String>) {
    val aa:List<String?> = listOf("A", "S", null, "D", "F")
    val qq = aa.mapIndexedNotNull({indx, v -> if (v==null)
{ null } else { "%d%s".format(indx, v)}})
    println(qq)
}
```

with the result being

```
[0A, 1S, 3D, 4F]
```

This is a pain as you have to handle nulls explicitly in the passed in function. So write an extension function on List called mapIndexedSkipNull which bypasses nulls in the list. The signature of this extension function should be

```
     fun <R,T> List<R?>.mapIndexSkipNull(transform:
(indx:Int, t:R) -> T? ) : List<T?>
```

Hence above you could replace “val qq …” with

```
     val qq = aa.mapIndexSkipNull(
                {indx, v -> "%d%s".format(indx, v)})
```

and get the identical result. Your extension function should be purely functional except that it may use a MutableList. (+1 extra credit for not using a mutable list)

4. Consider a 2 dimensional array; for example in Go it would be initialized as

```
    var arr2d [7][4]int8
```

Suppose that this array is actually stored in a language that uses row-major storage and that arr2d[1][3] is stored at 0x0100. (Recall that 0x0100 indicates a hexidecimal — base 16 — number.) At what location would arr2d[3][1] be stored. Explain.
PART 2: Alternately suppose that the array is stored in a language that uses column-major storage. If arr2d[1][3] is stored at location 0x0100 then where would arr2d[3][1] be stored. Explain, briefly.

5. Define polymorphism as it applies to programming languages. Describe how Kotlin and Go implement polymorphism.

6. Suppose there exists a language KotlinS that differs from Kotlin only in that it uses structural equivalence rather than name equivalence. Write a program that would compile and run in KotlinS but would do neither in Kotlin. Explain.

7.   Suppose there exists a programming language that has three modes of function parameters: pass-by-value (PBV), pass-by-reference (PBR) and pass-by-sharing (PBS). The syntax of the language is otherwise identical to Kotlin. Further suppose that the function parameter mode is set by a required first parameter for every function.   Still further suppose that in this programming language strings are mutable. So, for instance,

```
qqq(PBR, “hello”)
```

would call the function qqq in pass-by-reference mode and thus would give the function a reference to the string “hello”. Using Kotin syntax, the function defintion of qqq would be:

```
fun qqq(str: String)  { … }
```

That is, the parameter that controls the passing style is implicit in the definition of qqq. The function is affected by the mode but does not have direct access to the mode. Using mutable strings, write a single function in this language and some code to use that function that shows the differing effects of these three parameter modes listed above. Also explain what your function does and how it illustrates the differences among the parameter modes. Suggestion: use lots of print statements; then explain the output of each print.

8. Dangling references are a problem for a language that uses manual memory management. Can they also be a problem for a language that has automatic garbage collection? Explain.

9. Suppose that the CS department wanted to switch the Intro to CS and Data Structures courses from Java to either Kotlin (using imperative constructs) or Go. Which would you recommend? (You may not bail out and say “keep java”. Nor can you suggest some other language.) 2 points for each of the first 4 reasons that I consider valid. 1 points for a fifth reason.

10. Exceptions (in Java and Kotlin) are considered to be very “heavy” operations. That is, they are time consuming. Hence some classes provide a set of method calls that throw exceptions and an equivalent set of method calls with a special return value that indicates a problem has occurred (For instance, the Java Queue interface specifies the method `add` which throws an exception and `offer` which returns a special value). First, why are Exceptions heavy? Second, discuss why — other than heaviness — a programmer using a class that implements Queue might choose to use one version or the other.

11. Write code in Go that illustrates the use of currying. Also explain exactly why your code illustrates currying.