SENG2200/6220 –Programming Languages & Paradigms
Computer Lab for Week 2, Semester 1, 2018

Part 1 – Language Evaluation & Language Syntax Specification

1. What does it mean for a language to be "reliable/writeable/readable".

2. How does a good type system make a language reliable?

3. Write BNF that will define an identifier in Java.

4. Write a BNF specification that defines the syntax of a Java `while` loop.

5. Write a BNF specification that defines the syntax of a Java `if` statement.

6. Write a BNF specification that defines the syntax of a Java assignment statement.

Part 2 – Simple Java Revision

7. Draw a UML representation of a Student class. It should contain identifying information for the student and information on what courses the student is enrolled in.

8. Write an implementation of your Student class in Java. Concentrate on specifying the attribute data and the method prototypes.

9. Complete the constructor for your Student class from Q7/8.

10. Write Java code that will instantiate a Student object.

11. Write Java code that will declare and instantiate an array of Student(s) from Q8.

12. Write Java code that will populate the array of Q11.

13. Discuss the differences between what you have implemented in Java in Q8-12, and what you would write if similar was required to be implemented in C++.

14. What would be different (with respect to your Java array), if you wanted it to behave like a Stack.

Part 3 – Linked Lists in Java

15. In Java, there are no pointers - Java only uses references. Write a Node class in Java that is minimally sufficient for implementing Stack and Queue classes.

16. Write a Queue class in Java that will use your Node class from Q15. Decide on the attribute data required for your Queue class. It should implement the standard set of methods normally used as a public interface for a Queue class.

17. Expand your Node class of Q15 to allow Deque structures to be implemented.

18. Write a Deque class in Java, implementing the usual Deque interface.

19. Look at the all the places where your Deque class of Q18 needs to use *if* statements. Some of these can be removed by a slight change in design. How can this be done? What other methods need to change as a result? (Hint: not part of the interface.)

20. Alter your Deque class of Q19 so that it only uses a single sentinel node.