Answers to Midterm questions on 10/14/2014

OAKLAND UNIVERSITY, School of Engineering and Computer Science
Object Oriented Computing II
CSE 231 002/CSE 506 002

Chapter 1: (22 pts)

1. Waterfall
2. Unified Modeling language
3. B
4. Test bank chapter 1, 39 (2 points)

   The public access modifier allows access from anywhere. The Protected access modifier allows access from within the class, or within subclasses in the same package, or within subclasses in other packages. Package access allows access from within the class or within subclasses in the same package. Finally, private access only allows access from within the class.

5. A
6. B
7. Extends (extend (without s) will not get point)
8. Import
9. D
10. A
11. False ($O(N^3)$)
12. $O(N^4)$
13. $O(N^2)$
14. (6 points)
    best case: 2N
    worst case: 64N
    average: 32N
    O(N)

Chapter 2 (16 points)

15. Abstract Data Type or ADT

16. Application (or user or client) level. Logical (or abstract) level. Implementation (or concrete) level. Application level. 2 point question, Each level has 0.5 point

17. Precondition

18. constants (*final* variables) and *abstract* methods (2 points, 1 for each).

19. C,B,A,D,E [B, A, D] can mix in any order, but must be in between C and E. NO partial point for this one (2 points)
20. O(N)
21. A

22. It is not correct, the while loop will go infinite if it finds since it only update the found flag. There are several ways to correct it, e.g. put found as part of while terminatation condition (&& !found), or just do a return true when found. (4 points)

Chapter 6 (28 points)

23. C

24. compareTo

25. flag1=false; flag2 depends, if no equals is implemented, it is false. If equals is implemented. It still depends on how it is implemented (2 points)

26. Each element except the first has a unique predecessor, and each element except the last has a unique successor.

27. Interface

28. All objects on a sorted list support *compareTo*, since the only way for an object to be placed on a list is through the *add* operation and the *add* operation for a sorted list only accepts objects that are *Comparable*, i.e., that support *compareTo*, as stated in the pre-condition. (2 points)

29. The algorithm first finds the location in the list array where the target element should be put in. The if statement with compareTo will do this. The first loop is to find the location, and the second loop is the shifting up the elements after that location so the target element can be inserted. In this way, the order is still maintained. (4 points)

30. False

31. When looking for something in a sorted list first look at the middle element of the list and decide if you should continue searching in the front or back half of the list. Continue searching using this same approach, i.e., always looking at the middle element of the area you are currently search and cutting your search area in half. Binary search O($\log_2 N$), regular search O(N). (4 points)

32. (4 points)

method	efficiency
add	O(1)
remove	O(N)
size	O(1)
contains	O(N)

33. False

34. Previous = NULL: If the target equals to the first element of the list. there is no equal element, previous is the reference to the last element of the list, If there is equal element, previous point to the element in front of equal element. (6 points)

Chapter 7 (24 points):

35. In a circular linked list the last node is linked to the first node, whereas in a linked list the last node has a null link. (2 points)

36. False

37. Doubly Circular Linked list (or Circular Doubly linked list)

38. (6 points)

List type	Find A's previous node	Find A's next node
Linked list	O(N)	O(1)
Circular Linked List	O(N)	O(1)
Doubly Linked list	O(1)	O(1)

39. location.link.back = location.back (2 points), no partial

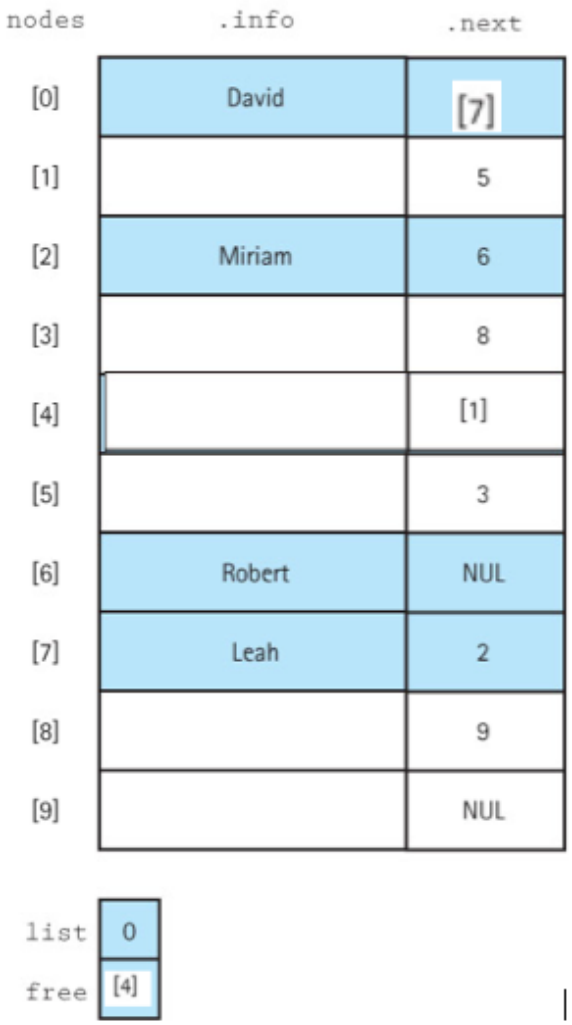
40. They can simplify processing by removing the need to code for special cases when dealing with the first or last nodes on a list.

41. index

42. Sometimes managing the free space ourselves gives us greater flexibility. There are programming languages that do not support dynamic allocation or reference types. There are times when dynamic allocation of each node, one at a time, is too costly in terms of time (3 points)

43. Add “private int back;” declaration to the AListNode class (2 points)

44. See below. If an answer put the freed node (4) linked after free node 9, (9.next = 4, and 4.next = NUL), it is also correct. (5 points)



Chapter 5 (15 points):

45. True

46. B

47. A

48. The one in the *BoundedQueueInterface* is defined as potentially throwing a *QueueOverfowException* while the one in the *UnboundedQueueInterface* is not. (2 points)

49. (6 points)

4 6 3

10
9
7
6

50. Implement the floating front design of array-based implementation of queue, i.e. the access to the array element wrapped around to the beginning of the array when it reach the end. (4 points)