Data Structures and Object Oriented Programming
Practice Problems for Midterm

Contents

1 Short answers

1. Given an array A = {54, 26, 93, 17, 77, 31, 44, 55, 20}, write the contents of the array after
   i) 2 passes (iterations) of insertion sort algorithm
   ii) 5 passes of the insertion sort algorithm
   iii) 7 passes of the insertion sort algorithm

2. Which of the following is not true about abstract classes?

   i) If we derive an abstract class and do not implement all the abstract methods, then the derived class should also be marked as abstract using 'abstract' keyword
   ii) Abstract classes can have constructors
   iii) A class can be made abstract without any abstract method
   iv)A class can inherit from multiple abstract classes

3. A deque (double ended queue) is a queue in which additions and deletions can be made from both ends or the queue. Assume we implement a deque as a Circular Singly Linked list with an external pointer pointing to the rearmost element of the queue. Which among the 4 operations : addFront, addRear, deleteFron and deleteRear is the most difficult to implement with this data structure? Why?

4. What does the acronym ADT stand for? What is an ADT? What is the connection, if any, between an ADT and a data structure? Use an example to illustrate the answer to your last question.

5. If the characters 'D', 'C', 'B', 'A' are placed in a Queue (in that order), and then removed one at a time, in what order will they be removed?

   a) A B C D
   b) D B C A
   c) D C B A
   d) A C B D

6. Which of the following is true about linked list implementation of stack?

   i)In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from end.
   ii) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from the beginning.
   iii) Both of the above
   iv) None of the above

7. Consider the following SecretMethod in the SinglyLinkedList class with head. (No dummy nodes)

```
public void SecretMethod() {

     if(head == null || head.getNext()==null)
          return;

     Node current = head;
     Node newHead = head.next();
     Node temp;

     while(current!=null && current.getNext()!=null) {
          temp = current.getNext();
          current.setNext(temp.getNext());
          temp.setNext(current);
          current = current.getNext();
     }
     head = newHead;
}
```

If the above method is called on a list 5, 9, 4, 12, 7, 21, 4 write the list contents after the method finishes execution.

2 Coding questions

1. Find out if a given Singly-Linked List terminates in null or has a cycle in it? (In a list with a cycle, the last node of the list may point to ANY other node in the list). *public boolean hasACycle()* { }
2. Reversing a Linked List: a) Reverse a given Doubly Linked List with head and tail nodes. b) Reverse a Singly Linked List with a head. Can you do it in one pass of the list?

   public void reverse() { }
3. Write a method called FindMid that returns the value of the node at the middle of a doubly linked list without using any form of counter. Assume that you are given the Node and DLList structures (no iterators) and their associated methods. Also, assume that the total number of nodes in the list is odd.
4. Write a method that removes all occurences of 'elem' in a Singly Linked List with head.

   public void removeAllOccurences(T elem) {
   }
5. Write a class, DualStack that implements two array-based stacks (stackA and stackB) using a single one dimensional array called DualArray[DEFAULT CAPACITY]. The array should be utilized in such a way that the stack overflow happens in either of

these two stacks only when the array itself is full.

You may write a constructor and initialize certain class variables, which you may feel necessary. Let topOfStackA and topOfStackB be the top indices for stackA and stackB respectively. This class should consist of only the following methods:

1. int size() : Sum of the sizes of stack 1 and stack 2.
2. void pushA(Object item) : Push the item into stackA
3. void pushB(Object item) : Push the item into stackB
4. Object popA() : Pop the top item from stackA
5. Object popB() : Pop the top item from stackB

Assume that the methods isEmptyA() and isEmptyB() exist for stackA and stackB, respectively, and isFull() exists. isFull() indicates when both stacks cover the entire array. You do not need to write the code for these methods but you may use them in writing the other methods.

Hint: the Stacks may grow in different directions in the array

6. Given a Singly Linked List with head, consisiting of 0s, 1s and 2s, write a method to sort the data in the LinkedList.

   Example: Input: $1->2->1->1->0->0->2$
   Output: $0->0->1->1->1->2->2$

7. Write a method isIdentical that checks whether two LinkedLists are identical. Two lists are identical if they have the same contents. Assume you have access to the head. Each node has a data and a next variable. (and prev)

   public boolean isIdentical(DoublyLinkedList12 list1, DoublyLinkedList12 list2)

Extra Credit Problems from HW2

8. Given a Doubly linked list with head and tail references, rotate the linked list counter-clockwise by k nodes, where k is a given positive integer. For example, if the given linked list is $10->20->30->40->50->60$ and k is 4, the list should be modified to $50->60->10->20->30->40$.

9. Given two DoublyLinkedLists with integer data, write a method in DoublyLinkedList12 class that populates a third list with the sum of the numbers represented by the two lists. Also provide a few JUnit tests in a separate file.

   Example 1:
   $Input1 : 1->2->3$

$Input2 : 4- > 5- > 6$
$Result : 5- > 7- > 9$

$Example2 :$
$Input1 : 1- > 1- > 1- > 1$
$Input2 : 9- > 9$
$Result : 1- > 2- > 1- > 0$

Method signature: *public void sumOfTwoLists(DoublyLinkedList12*$< Integer > list2, DoublyLinked$
$Integer > result)$