NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

MIDTERM TEST FOR
Semester 2, AY2018/19

CS2040 – Data Structures and Algorithms

March 2019 Time allowed: 1.5 hours

STUDENT NO. :	A	0	1						

(Write your Matriculation Number legibly with a pen.)

INSTRUCTIONS TO CANDIDATES

0. Write your student number in the space provided above.
1. This paper consists of 10 MCQ questions, and 2 structured questions. It comprises ten (10) printed pages including this front page.
2. Answer all questions. Use the provided OCR form for the 10 MCQ questions, and write your answer for the short questions directly in the space given after each question.
3. Use 2B or darker pencil to shade the OCR form. You are allowed to use PENCIL to write your answers in this question paper.
4. You must submit both the OCR form and this paper. It is your responsibility to ensure that you have submitted both to the invigilator at the end of the test.
5. If you do not write/shade the student number or the student number is incorrect, we will consider it as you did not submit your answers.
6. No extra time will be given at the end of the test for you to write/shade your student number and transfer your MCQ answers to the OCR form. You must do all these before the end of the test.
7. Marks allocated to each question are indicated. Total marks for the paper is 100.

Question	***Max***	***Marks***
Q1 – Q10	30	From OCR
Q11	15	
Q12	55	
Total	**100**	

Section A: 10 MCQ (30 Marks) Shade the correct answer on the OCR form provided.

1. Which of the following is/are **true about ADT**?

   i) We cannot create an ADT if the class name is already available in Java API
   ii) We use interface to tell the client program what can be done with the ADT
   iii) ADT is used to achieve encapsulation in OOP programming.

   a) i) only
   b) ii) only
   c) iii) only
   d) ii) and iii) only
   e) None of the above choices (a) – (d)

2. What is the complexity of the following segment of codes? Take note of the complexity of the methods in the underlying data structure.

```
ArrayList <Integer> myList = new ArrayList <Integer>();
for (int i=0; i < n; i++)
   if (myList.indexOf(i) == -1)
       myList.addFirst(i);
```

   a) O(n)
   b) $O(n^2)$
   c) $O(n^3)$
   d) $O(n^4)$
   e) None of the above

3 Which of the following statement(s) is/are **TRUE**?

   i) A circular linkedlist must have a head pointer and a tail pointer.
   ii) The Linkedlist class in Java API implements the ListInterface.
   iii) You must use an Iterator to make the LinkedList in Java API a doubly linkedlist.

   a) i) only
   b) ii) only
   c) i) and ii) only
   d) ii) and iii) only
   e) None of the above choices (a) – (d)

4. Which of the following is the postfix expression of (a + b – c * d) / e – ((f + g) * j)

   a) abcdefgj*+-/*-+
   b) ab+c-d*e/f+gj*e-
   c) ab+cd*-e/fg+-j*
   d) abc+d*-e/fg+j*-
   e) ab+cd*-e/fg+j*-

5. Which of the following recursive methods does not satisfy the 3-finger rules?

i)
```
void M1(int n) {
   if (n/2 <= 0) return;
   System.out.println(n + M1(n/2));
}
```

ii)
```
boolean M2(string str) {
   if (str.length() >= 2) {
      if (str.charAt(0) == str.charAt(str.length-1))
           M2(str.subString(1,str.length-1));
      else return true;
   else return false;
}
```

iii)
```
/* pre-cond: curr is not null before first call */
ListNode M3(ListNode curr) {
    if (curr.next == null) return curr;
    else return M3(curr.next.next);
}
```

a) M1 only
b) M2 only
c) M3 only
d) M1 and M3
e) M2 and M3

6. Which of the following statement(s) is/are **TRUE**?
   i) We can use two stacks to simulate a queue
   ii) We can use two queues to simulate a stack
   iii) Using two stacks to do sorting is similar to insertion sort

a) i) and ii) only
b) ii) and iii) only
c) i) and iii) only
d) all three statements
e) none of the statements

7. What is the big-O for the following recursive method? Read **carefully**.

```
public void what_is(int n)   {     // n is large
    if (n > 1) {
        System.out.println("first call: ");
        what_is (n/2);
        for (int i = 0; i < n; i++) {}
            for (int j = n; j > 0; j--)
                System.out.println(n*n + " is n^2");
        System.out.println("second call: ");
        what_is(n/2);
    }
}
```

a) O(log n) b) O(n) c) O(n log n) d) O(n^2) e) None of the above

8. Given a bank account object with 2 attributes - account holder name and balance in account, and an array of 10 such objects as follows

Name	Mary	John	Tom	Abigail	Emmalyn	Hannibal	Evelyn	Martha	Zechariah	Appleby
Balance	12345	200000	20	472798	13333333	20	77834	9389	357683	77834

What does the array look like after we sort (non-decreasing order) first by name then by balance using **merge sort**?

a)

Name	Abigail	Appleby	Emmalyn	Evelyn	Hannibal	John	Martha	Mary	Tom	Zechariah
Balance	20	20	9389	12345	77834	77834	200000	357683	472798	13333333

b)

Name	Abigail	Appleby	Emmalyn	Evelyn	Hannibal	John	Martha	Mary	Tom	Zechariah
Balance	472798	77834	13333333	77834	20	200000	9389	12345	20	357683

c)

Name	Hannibal	Tom	Martha	Mary	Appleby	Evelyn	John	Zechariah	Abigail	Emmalyn
Balance	20	20	9389	12345	77834	77834	200000	357683	472798	13333333

d)

Name	Tom	Hannibal	Martha	Mary	Appleby	Evelyn	John	Zechariah	Abigail	Emmalyn
Balance	20	20	9389	12345	77834	77834	200000	357683	472798	13333333

e)

Name	Hannibal	Tom	Martha	Mary	Evelyn	Appleby	John	Zechariah	Abigail	Emmalyn
Balance	20	20	9389	12345	77834	77834	200000	357683	472798	13333333

9. If the partition function of quicksort is modified to be as follows

```
private static int partition(int[] a, int i, int j) {
  int pivot = a[i];
  int[] s1 = new int[j-i+1];
  int[] s2 = new int[j-i+1];
  int left = 0;
  int right = 0;

  for (int k=i+1; k<=j; k++) {
    if (a[k] < pivot)
      s1[left++] = a[k];
    else
      s2[right++] = a[k];
  }
  for (int l=0; l < left; l++)
    a[i+l] = s1[l];

  a[i+left] = pivot;

  for (int l=0; l < right; l++)
    a[i+left+l+1] = s2[l];

  return i+left;
}
```

Which of the following statement(s) is/are **TRUE**?

i. This modified quicksort will not sort correctly in general
ii. This modified quicksort will have time complexity worse than O(NlogN) in the best case
iii. This modified quicksort will still have time complexity O(NlogN) in the best case
iv. This modified quicksort will not be an in-place sort
v. This modified quicksort will be a stable sort

a) i only
b) ii and iv only
c) ii and v only
d) iii and iv only
e) iii and iv and v only

10. Richard runs a factory which manufactures customized boxes of different weights. For each customer, Richard will get his workers to sort the **N** (10,000 <= **N** <= 1,000,000) boxes made for the customer by non-decreasing order of weight. However due to the volume of boxes manufactured, the workers will start to make mistakes in the sorting with the last **C** (**C** <= 100) number of boxes - meaning those c boxes will be placed in the wrong position. So Richard has decided to ask you to write a program to re-sort the boxes using as input the current wrong ordering given by his workers. Which of the following would be the best sorting algorithm to use in this case?

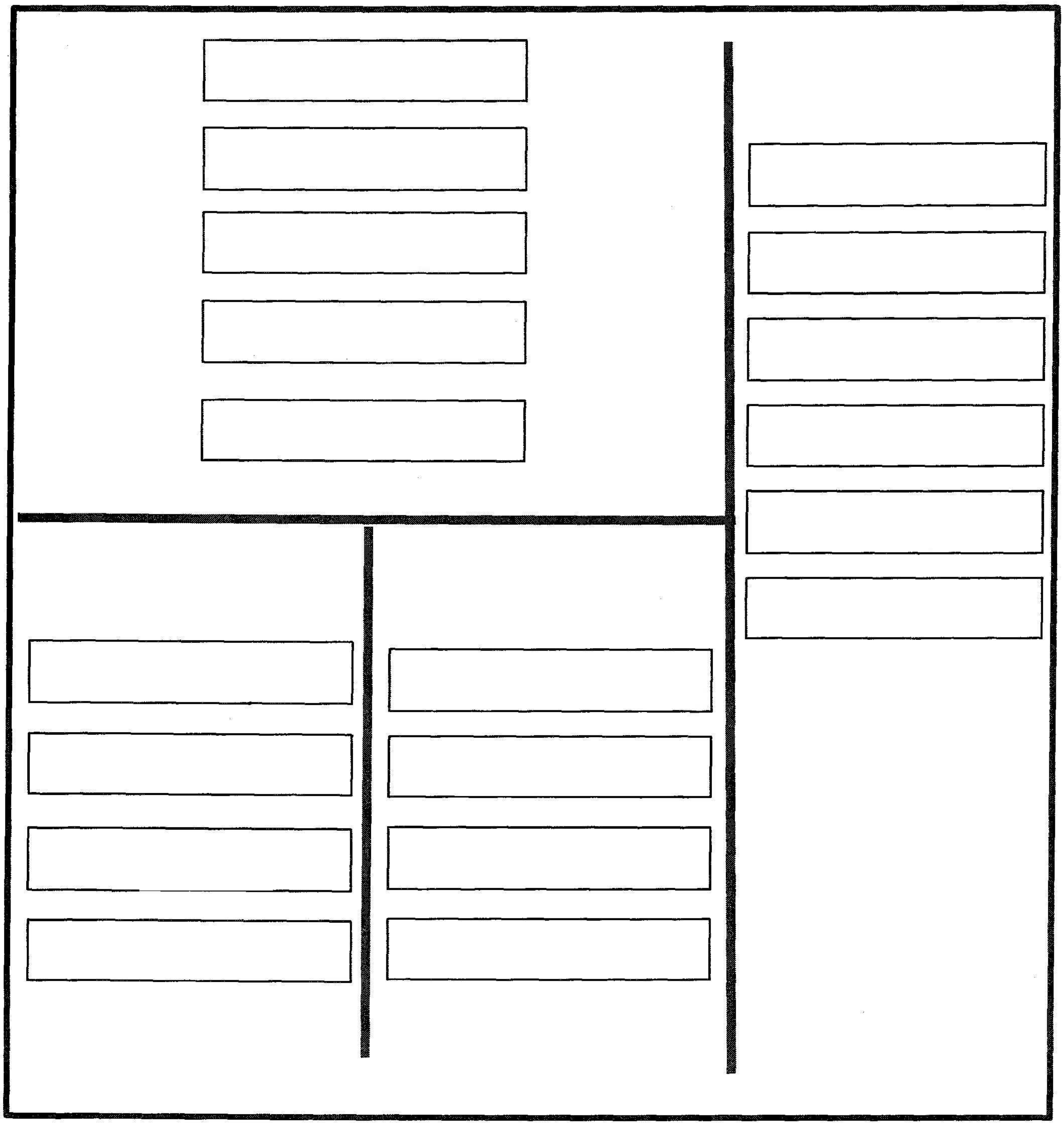
a) Selection Sort
b) Insertion Sort
c) Bubble Sort
d) Merge Sort
e) Quick Sort

Section B: Structured problems (70%)

Q11. As mentioned in the lectures, it is important to design a complete algorithm before coding the program. We also use one horizontal line and two vertical lines to help us decide what should be done during the design process.

Put the following list of words/phrases in the **MOST** relevant boxes below. (15 marks)

Driver class, Implementation, Scanner class, ADT, Java API, Specification, Interface, Application, input/output, Math class, Comparable, create objects, ExtendedLinnkedList.java, main method, ArrayList,



Q12. You may use the ListNode class and the TailedLinkedList specification given on the last page of this paper for this questions.

a) Given an unsorted TailedLinkedList of integers, write an algorithm to separate the integers into two **TailedLinkedLists** where the first list stores all the even integers in the same order as the original list and the second list stores all the odd integers in the opposite order as the original list. Note that you are **NOT** allowed to use stack and queue in your solution. (20 marks)

For example 5 -> 2 -> 7 -> 3 -> 4 -> 6 -> 9

will produce 2 -> 4 -> 6 (even integers in the same order as in original list) and

the second list is created with 9 -> 3 -> 7 -> 5 (odd integers in the opposite order as the original list)

Note that your algorithm must be so detail that we are able to do a 1-to-1 translation to Java

b) Given an unsorted TailedLinkedList of integers, write an algorithm to create two **circular** LinkedLists where the first list stores all the even integers in the same order as the original list and the second list stores all the odd integers in the opposite order as the original list. You are **NOT** allowed to create new node but you are allowed to use Stack and Queue in your solution. (20 marks)

Note that your algorithm must be so detail that we are able to do a 1-to-1 translation to Java

c) Given a TailedLinkedList of integers, write a recursive algorithm to reverse the linkedlist. No additional data structure should be used and you should not create new node in your solution.

(15 marks)

```
class ListNode {
        private int _value;
        private ListNode _next;
        public ListNode (int value) {_value = value;    _next = nil;}
        public ListNode (int value, ListNode next) { _value = value;  _next = next;}
        public int getValue () {return _value;}
        public ListNode getNext() {return _next;}
 }

 interface TailedLinkedList {
        public Boolean isEmpty();     // check if the list is empty
        public int size();            // return size of the list
        public int getFirst();        // return the value in the first node
        public void addFirst(int value); // add the value as the first node in the list
        public int getLast();            // return the last integer in the list
        public void addLast(int value);  // add value as the last item in the list
        public void addAfter(ListNode curr, int value); // add value after the node pointed by curr
        public int removeAfter(ListNode curr); // remove the item after the node pointed by curr
 }
```

-- End of Paper --