SECTION A

SAMPLE A 2001

PAPER CS210

ALGORITHMS AND DATA STRUCTURES I

Dr. R. Procter, Mr. S. Brown, Mr. T. Naughton.

Answer ALL QUESTIONS from Section A.I and any TWO questions from Section A.II.

Negative marking will be applied for Section A.I (4 for a correct answer, –1 for an incorrect answer, 0 for no attempt).

Time Allowed: 1.5 hours.

SECTION A.I (60 marks)

1 Searching an array for an element that is definitely in the array requires $(n-1)$ comparisons in the worst case, where n is the length of the array. How many comparisons in the worst case if the element might not be in the array? [4 marks]

(a) 1
(b) 2
(c) $n-1$
(d) $n-2$
(e) none of the above

2 If an algorithm has a complexity of $2(\log_2 n - 3) + 2(n^2 - 1)$ for some model of computation (some set of assumptions) and some complexity measures (such as number of comparison operations) we could say that it has complexity [4 marks]

(a) $= O(6n)$
(b) $= O(n^2)$
(c) $= O(2 \log_2 n + 6n)$
(d) $= O(\log_2 n)$
(e) none of the above

3 Inserting an element into an unsorted array of length n that is not yet full requires how many comparison operations between elements in the worst case (assume that you do not know how many elements are in the array)? [4 marks]

(a) $O(\log_2 n)$
(b) $O(n)$
(c) $O(1)$
(d) $n^2 - 1$
(e) none of the above

4 Inserting an element into a sorted array of length n that is not yet full requires how many comparison operations between elements in the worst case? Assume the number of elements in the array is known. [4 marks]

(a) $O(\log_2 n)$
(b) $n - 1$
(c) $O(1)$
(d) $n^2 - 1$
(e) none of the above

5 Finding and removing an element from a sorted array of length n requires how many copy operations and how many comparisons between elements, in the worst case (assume to remove an element one copies all subsequent elements down one position)? [4 marks]

(a) $\lceil \log_2 n \rceil$ copy operations and n comparisons between elements
(b) $O(1)$ copy operations and n comparisons between elements
(c) $O(n)$ copy operations and $\lfloor \log_2 n \rfloor + 1$ comparisons between elements
(d) $O(n)$ copy operations and n comparisons between elements
(e) $n - 1$ copy operations and $n - 1$ comparisons between elements

6 Searching a stack of size n (whose elements are sorted in ascending order) for a particular element will require how many `pop()` operations, in the worst case. [4 marks]

(a) $n - 1$
(b) $O(n)$
(c) $\lfloor \log_2 n \rfloor + 1$
(d) $n^2 - 2$
(e) none of the above

7 Inserting an element into an array of length n that is known to be full requires how many copy operations and how many comparisons between elements in the worst case? (Assume you must add it to the array and cannot simply return an error message.) [4 marks]

(a) $\lceil \log_2 n \rceil$ copy operations and $O(n)$ comparisons between elements
(b) $O(n)$ copy operations and $O(1)$ comparisons between elements
(c) $O(1)$ copy operations and $\lfloor \log_2 n \rfloor + 1$ comparisons between elements
(d) $O(n)$ copy operations and $O(n)$ comparisons between elements
(e) $n + 1$ copy operations and n comparisons between elements

8 Adding an element to an unsorted array of length n, that does not allow duplicates, requires how many comparisons between elements in the worst case? Assume that the array is not yet full and the number of elements in the array is known. [4 marks]

(a) $n - 1$ comparisons between elements
(b) n comparisons between elements
(c) $\lceil \log_2 n \rceil$ comparisons between elements
(d) $n/2$ comparisons between elements
(e) $\lfloor \log_2 n \rfloor + 1$ comparisons between elements

9 Adding an element to a sorted array of length n, that does not allow duplicates, requires how many comparisons between elements in the worst case? Assume that the array is not yet full and the number of elements in the array is known. [4 marks]

(a) $n - 1$ comparisons between elements
(b) n comparisons between elements
(c) $\lceil \log_2 n \rceil$ comparisons between elements
(d) $\lceil n / 2 \rceil$ comparisons between elements
(e) $\lfloor n / 2 \rfloor + 1$ comparisons between elements

10 Searching a queue of `ints` for an element that may or may not be in it will require n `dequeue()` operations in the worst case, where n is the number of elements in the queue. If you know in advance that all elements in the queue have only one of two possible values (0 or 1), the number of `dequeue()` operations will (in the worst case) be [4 marks]

(a) $\lceil n / 2 \rceil$
(b) $\lfloor n / 2 \rfloor + 1$
(c) $\lceil \log_2 n \rceil$
(d) 2
(e) none of the above

11 Is it possible to perform a binary search on a linked list of `Comparables`? [4 marks]

(a) yes, because the list can be sorted
(b) no, because the `Comparables` could reference abstract data types
(c) yes, because the length of the list can be obtained
(d) no, because the list must be searched sequentially
(e) yes, because `Comparables` have a `CompareTo()` method

12 What is the output of the following tester program for `MyQueue` and `MyStack`? [4 marks]

```
MyQueue q;
MyStack s;
q = new MyQueue();
s = new MyStack();
s.push(new Integer(5));
s.push(new Integer(6));
s.push(new Integer(7));
System.out.print(s.pop());
q.enqueue(s.pop());
q.enqueue(new Integer(5));
q.enqueue(new Integer(6));
System.out.print(q.dequeue());
s.push(q.dequeue());
System.out.print(s.pop());
System.out.print(s.pop());
```

(a) 7655
(b) 7556
(c) 7565
(d) “Exception: cannot pop(), stack is empty”
(e) none of the above

13 What is the output of the following tester program for `MyQueue` and `MyStack`? [4 marks]

```
MyQueue q;
MyStack s;
q = new MyQueue();
s = new MyStack();
s.push(new Integer(5));
s.push(new Integer(6));
q.enqueue(s.pop());
q.enqueue(new Integer(5));
q.enqueue(new Integer(6));
q.enqueue(new Integer(7));
System.out.print(q.dequeue());
while (!q.isEmpty()) {
  s.push(q.dequeue());
}
System.out.print(s.pop());
System.out.print(s.pop());
```

(a) 667
(b) 676
(c) 765
(d) 656
(e) none of the above

14 What will be the result of executing the following function when, initially, *a* is an array of length 4 of `Integers` with data values {1, 2, 1, 3} and *b* is an `Integer` with data value 1? [4 marks]

```
public static void example(Object a[], Object b) {
  /* This is not a useful comment.
  */
  int count; // a counter
  count = 0;
  while (count < a.length) {
    if (!a[count].equals(b)) {
      System.out.print(a[count].toString());
      a[a.length-count-1] = b;
    }
    count++; // this is equivalent to "count = count + 1"
  }
}
```

(a) "23" will be printed to the screen
(b) "213" will be printed to the screen
(c) *a* will be {1, 1, 1, 1}
(d) *a* will be {1, 2, 2, 3}
(e) none of the above

15 What is the worst-cost complexity of the following algorithm. Model: assume that variables take up zero space, that assignment and print statements take one unit of TIME each, and that all other statements and operations take zero TIME. Measures: measure TIME. [4 marks]

```
public static void example(Object a[], Object b) {
  /* This is not a useful comment.
  */
  int count; // a counter
  count = 0;
  while (count < a.length) {
    if (!a[count].equals(b)) {
      System.out.print(a[count].toString());
      a[count] = b;
    }
    count++; // this is equivalent to "count = count + 1"
  }
}
```

(a) $\log_2 n + 2$, where n is the length of the array
(b) $(3/2)n + 1$, where n is the length of the array
(c) $O(n^2)$, where n is the length of the array
(d) $3n + 1$, where n is the length of the array
(e) none of the above

SECTION A.II (40 marks)

1 Write a method called `findSubseq()` that takes an `int` *a*, an `int` *b*, and an array of `ints` *c*. The method returns the index of the first occurrence of *b* consecutive copies of *a* in the sequence of integers *c*. If *a* does not occur consecutively *b* times in *c* then return `-1`. [20 marks]

```
int findSubseq(int a, int b, int c[]) {
  /* Return the index in c of the first occurrence of b
  ** consecutive copies of a. Otherwise return -1.
  ** Assume c will not be null.
  */
}
```

For example, if *a*=5, *b*=2, and *c*={1,7,5,5,5,2} the method would return 2. If *a*=5, *b*=2, and *c*={3,5,7,3} the method would return `-1`. For simplicity, you may assume that `b>0` and `c.length>0`.

2 Write a function that takes an `int` queue *q* as an argument, and returns a queue containing (in reverse) every third element of *q* starting with the last element. For example, if q contains {a, b, c, d, e} then a queue containing {e, b} should be returned. [20 marks]
You may only declare `int`, `MyStack`, and `MyQueue` variables. `MyQueue` has methods `clear()`, `enqueue()`, `dequeue()`, and `isEmpty()` only. `MyStack` has methods `clear()`, `push()`, `pop()`, and `isEmpty()` only. Assume that the stack and queue can never be overfilled.

```
MyQueue example(MyQueue q) {
  /* This function returns a queue containing the reverse of
  ** every third element of q.
  */

  // Your code here
}
```

```
public class NodeA {
  /* This class defines a node for a linked list of Comparables.
  ** The node has one reference and one link.
  */
  private Comparable data;
  private NodeA next; // link to next node

  public NodeA(Comparable newData) {
    /* Conversion constructor. Link is set to null.
    */
    data = newData;
    next = null;
  }
  public Comparable getData() {
    /* Return a reference to the data at this node.
    */
    return data;
  }
  public NodeA getNext() {
    /* Return a reference to the next node in the list.
    */
    return next;
  }
  public NodeA setNext(NodeA nextNode) {
    /* Set reference to the next node in the list.
    */
    next = nextNode;
  }
}
```

Figure 1. The `NodeA` class.

3 Write a method `mergeLists()` that is passed two linked lists *a* and *b* of `NodeA` objects. (The `NodeA` class contains a single `Comparable` and is given in Figure 1.) Neither list *a* or *b* contains duplicate values. Both lists are sorted in ascending order. The method should return a new linked list that contains copies of references to all of the elements of *a* and *b* (without duplicates) and is also sorted in ascending order. Empty lists are passed as `null`. [20 marks]

If you call any other methods from your `mergeLists()` method you must supply their code.