Esc101N, Jan-Apr 2005, Second Mid-Semester Exam ANSWER KEY
Fri 11 Mar 2005, Duration: 1 hour (14:00-15:00), Total marks: 150

Name: BHASKARAN RAMAN **Roll No:** _____ **Section:** ____

Note: This exam is open hand-written (your own) notes. No xerox copies, books, other printed material, or electronic material is allowed. Please answer in the space provided for each question. You may refer to the brief notes given in the last page.

Question 01: Follow the Recursion [5+6+9=20 marks] – Consider the following code:

```
class Q1 {
      public static void rec_compare(int[] A, int i) {
            if(i == A.length-1) { return; }
            System.out.println(A[i] < A[i+1]); // statement S1
            rec_compare(A, i+1); // statement S2
      } // End rec_compare()
      public static void main(String[] args) {
            int[] A = { 5, 3, 8 }; // statement S3
            rec_compare(A, 0);
      } // End main()
} // End class Q1
```

(a) What is printed by the above program?

false
true

→ 5 marks for correct answer
Can be lenient about "True" instead of "true" and such variations

(b) Suppose the order of statements S1 and S2 is changed, what would be printed?

true
false

→ 6 marks for correct answer

(c) Change the statement S3 such that the printed output would be the same in both cases (a) and (b).

9 marks for correct answer

`int[] A =` {5,5,5} ;

→ any increasing, or decreasing, also will work.
new int[3] will also work.

Question 02: User-Friendly Input [15 marks] – The following method reads the week day as input from the user. The user is supposed to enter one of the strings `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat`, or `Sun`. The method prints an error message on a wrong input, and asks the user again. The method is supposed to return 0 for Mon, 1 for Tue, … 6 for Sun. Assume that `br` is a BufferedReader object variable which has already been initialized appropriately. Fill in the blanks below.

```
public int readWeekDay() throws IOException {
    for(;;) { // Appropriate loop
        System.out.print("Enter week day: "); // print prompt
        String input = br.readLine();
        if(input.equals("Mon")) { return 0; }
        if(input.equals("Tue")) { return 1; }
        if(input.equals("Wed")) { return 2; }
        if(input.equals("Thu")) { return 3; }
        if(input.equals("Fri")) { return 4; }
        if(input.equals("Sat")) { return 5; }
        if(input.equals("Sun")) { return 6; }
        System.out.println("Enter valid weekday"); // print error msg
    } // End loop
} // End readWeekDay()
```

→ any infinite loop

→ any such similar prompt

Deduct marks for any other sophisticated loop logic (more extra lines of code ⇒ deduct more marks)

3 marks total, 3 marks each for the other blanks

→ or any such similar message

Question 03: Referring to Each Other [2+3x6=20 marks] – Complete the following code such that, in the main method, the following box-pointer diagram results after the execution of the two statements.

2 marks

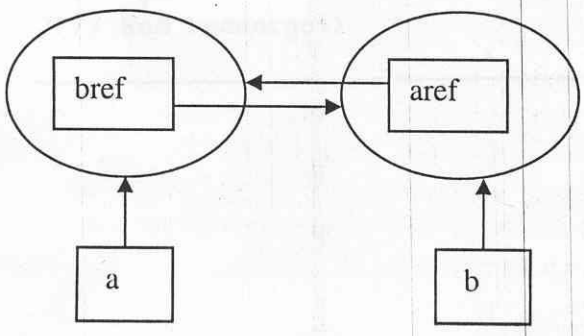
```
class A {
    public B bref;
    public A() {
        bref = null;
    }
} // End class A
```

3x6 marks

```
class B {
    public A aref;

    public B(A aref) {
        this.aref = aref;
        aref.bref = this;
    } // End B()

    public static void main() {
        A a = new A();
        B b = new B(a);
    } // End main()
} // End class B
```



I do not know of any other valid solution.
If there is any, it is your discretion.

Question 04: Recursive Merge [70 marks] – The following method is intended to merge two sorted (in increasing order) arrays A and B into a third new array and return the new array. The method `merge` uses the recursive method `recmerge`. Fill in the blanks.

```
public static int[] merge(int[] A, int[] B) {
      int[] C = new int[A.length+B.length];
      recmerge( A , 0 , B , 0 , C , 0 );
      return C;
} // End merge()

public static void recmerge( int[] A , int aIndex , int[] B ,
                             int bIndex , int[] C , int cIndex ) {
  if( cIndex == C.length) { return; } // base case
  if( aIndex == A.length) {
    C[ cIndex ] = B[ bIndex ];
    recmerge( A , aIndex , B , bIndex+1 , C , cIndex+1 );
  } else if( bIndex == B.length) {
    C[ cIndex ] = A[ aIndex ];
    recmerge( A , aIndex+1 , B , bIndex , C , cIndex+1 );
  } else if(A[ aIndex ] <= B[ bIndex ]) {
    C[ cIndex ] = A[ aIndex ];
    recmerge( A , aIndex+1 , B , bIndex , C , cIndex+1 );
  } else {
    C[ aIndex ] = B[ bIndex ];
    recmerge( A , aIndex , B , bIndex+1 , C , cIndex+1 );
  }
} // End recmerge()
```

2 x 6 ←

2 x 6 ←

2 marks ← if(cIndex == C.length) { return; } // base case

4 marks ← if(aIndex == A.length) {

2 marks ← C[cIndex] = B[bIndex];

6 marks ← recmerge(A , aIndex , B , bIndex+1 , C , cIndex+1);

4 marks ← } else if(bIndex == B.length) {

2 marks ← C[cIndex] = A[aIndex];

6 marks ← recmerge(A , aIndex+1 , B , bIndex , C , cIndex+1);

4 marks ← } else if(A[aIndex] <= B[bIndex]) {

2 marks ← C[cIndex] = A[aIndex];

6 marks ← recmerge(A , aIndex+1 , B , bIndex , C , cIndex+1);

} else {

2 marks ← C[aIndex] = B[bIndex];

6 marks ← recmerge(A , aIndex , B , bIndex+1 , C , cIndex+1);

→ order of args can be different but has to be consistent

- **Question 05: Median [10+15=25 marks]** – Given an array of unique numbers, find the median. The median of a set S of unique numbers is a number x such that:
  - x is a member of S
  - Suppose n1 is the number of elements in S <= x, and n2 is the number of elements in S > x. Then abs(n1-n2) <= 1.

This definition is actually ambiguous. Award marks if the code has been written to conform to this definition, OR 5 marks if the student has corrected the ambiguity. 5 marks

(a) Complete the following code to find the median of an array A of unique numbers.

```
public static double median(double[] A) {

    for(int i = 0; i < A.length; i++) {

        // Code for counting the number of elements <= A[i]
        int numLEAi = 0;
        for(int j=0; j< A.length; j++) {
            if (A[j] <= A[i]) { numLEAi++; }
        } // End for(j)

        // Finish up the code
        int numGAi = A.length - numLEAi;
        if (Math.abs(numGAi - numLEAi) <= 1) { return A[i]; }
    }

} // End median
```

Other similar solutions are acceptable so long as the logic is correct.

(b) We are interested in counting the number of times we have to compare one double with another, in the above code. For a given (random) array A, what is the minimum number of such "double" comparisons required in the above code? What is the maximum number of such comparisons? Give your answer in terms of A.length.

8+7=15

min # comparison reqd = A.length

max # comparisons reqd = A.length * A.length

Notes:

- If x is an object of class BufferedReader, "appropriately" initialized, x.readLine() returns a String as typed by the user
- The method main can be used without the String[] args

Good Luck!
