RMIT University

Programming 1 (COSC1073/COSC2362) Semester 2, 2012—Mid-Semester Test

A) Student Name (in full):	*Student Signature(11 September 2012)*
Student ID:	

Part I Multiple Choice / Fill in the blanks   30 marks

Select the most appropriate response and circle the corresponding letter (A/B/C/D/E/F). There is only one correct answer for each question.

1. **(6 Marks) The expression** `8+14  /  5*3;` `evaluates to`

A) 1
B) 8.4
C) 14
D) 12
E) 17

2. (6 Marks) What are the values of the specified variables, `a` to `e`, after all of the following statements have been executed.

```
int a, b, c, d, e;a = 7

                         b = 3
                         c = 2
                         d = 4
                         e = a

                         a = b
                         b = e
                         e = c
                         c = d
                         d = e
```

B) a = 1, b = 2, c = 3, d = 4, e = 5
C) a = 3, b = 7, c = 2, d = 3, e = 2
D) a = 3, b = 7, c = 4, d = 2, e = 2
E) a = 3, b = 7, c = 3, d = 2, e = 2
F) None of the above, the correct answer is:

a = 3, b = 7, c = 4, d = 2, e = 2

3. (6 Marks) A class called `Account` has a mutator (setter) method called `transfer` (…) to transfer an amount of money bal into account acc. Consider the following declarations and statement.

Account mum = new Account("MUM", 1000.00);
Account dad = new Account("DAD", 500.00);
if (mum.transfer(dad, 200) == true)
  ….

Which of the following statements best represents a correct method design for this purpose?

A) public void transfer(Account acc, double amount)
B) private boolean transfer(Account acc, double amount)
C) public boolean transfer()
D) public boolean transfer(double amount, Account acc)
E) private void transfer()

4. (6 Marks) What will the method `foo` return after it is called the second time?

```
public static void main(String[] args) {
     for (int i = 2; i < 7; i += 2) {
           System.out.println(foo(i));
     }
}
public static int foo(int a) {
     if (a - 1 > 3) {
           return 5 * a - 1;
     }
     else {
           return a + 1;
     }
}
```

A) 4
B) 2
C) 5
D) 9

5. (6 Marks) The class `Rectangle` is used to represent rectangles. An example of an instance of this class is `Rectangle(10, 20, 3, 5)`, representing a rectangle of width 3 and height 5, and with the bottom left-hand corner at coordinate (10, 20). Consider the following code fragment.

```
Rectangle r1 = new Rectangle(10, 20, 3, 5);
Rectangle r2 = new Rectangle(10, 30, 40, 50);
Rectangle r3 = new Rectangle(1, 2, 50, 40);
r3 = r1;
r1 = r2;
r2 = r3;
```

After the code is executed, which one of the statements that follow is true?

A) `r1` and `r2` point to two instances of `Rectangle` with attributes (10, 20, 3, 5).
B) `r1, r2` and `r3` point to the same instance of `Rectangle` with attributes (1, 2, 50, 40).
C) `r1` and `r2` point to the same instance of `Rectangle`.
D) `r1` and `r3` point to the same instance of `Rectangle`.
E) None of the above.

Part II (Program segments) 35 Marks
Write your answers in the space provided

6. (12 Marks) Complete the following code fragment by using an appropriate loop to step through and print 20 terms of the Fibonacci series, beginning with the terms 8 and 13. In the Fibonacci series each term is the sum of the previous two terms. Hence, the first few terms starting with 8 and 13 will be 8 13 21 34 55 89 …..

```
int term1 = 3;

int term2 = 5;

int printT;

for (int k = 1; k <= 20; k++)

{

    printT = term1 + term2;

    System.out.print(printT);

    term1 = term2;

    term2 = printT;

}

System.out.println();
```

Marking Guide

+ 6 marks for ensuring that 20 numbers are generated (no matter what the numbers are) and it does not matter if these are listed on one line or separate lines, because the question does not clearly say specify this. Deduct 2 marks for each of incorrect initialisation; loop condition; increment (this should apply to a while or do-while as well).
+ 2 marks for print statement to display each term
+ 4 marks for code to correctly generate the right sequence of numbers, starting from 8

7. **(11 marks)** Student marks are stored in the array **m** (unspecified size). Complete the missing part of the program below to find and print the average mark to one decimal place.
Hint: You may assume that every element of the array is populated, so you can use `m.length` (the array length for array m) to represent the total number of marks held in the array.

```
public class FindAverage {
  public static void main (String[] args) {
    int m[] = {28,56,94,56,94,30,28,28,30,94,94, … };
    // Declare variables "sum" and "average" to store
    // the sum of the marks and the resultant average mark


    int sum = 0;

    double average;

// The remaining code goes here

    for (int k = 0; k < m.length; k++)

    {

        sum += m[i];

    }

    average = (double) sum / m.length;

    System.out.printf("The average = %.1f\n", average);

  }
}
```

Marking Guide

+ 2 marks for declaration of "sum" and its initialisation or assignment to 1
+ 1 mark for declaring "average" as double (0.5 if declared as int)
+ 2 marks for correct loop to span all numbers in array m
+ 2 marks for correct computation of sum
+ 2 marks for correct average computation (if average was declared as int, ignore that here, but mae sure that the int divide is accounted for, either via a cast or some other means; if int divide is not accounted for then deduct a mark)
+ 2 marks for print statement to display average to one decimal place

8. (12 marks) The following block of Java code prints a 10x10 triangle of asterisks - the output of which is depicted below and referred to as the 'original triangle'. Variable `size` holds the value 10.

```
for (count = 0; count < size; count++) {
    for (int i = 0; i < count; i++)   // NOTE: Forgot int i in
                                      // original question
      System.out.print(" ");
    for (int k = 0; k < size - count; k++)

     System.out.print("*");

    System.out.println(); // NOTE: Forgot this in original question

}
```

The output would be:

```
**********
 *********
  ********
   *******
    ******
     *****
      ****
       ***
        **
         *
```

(a) Write a block of code to print the original triangle flipped ***vertically*** with an output of:

```
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
**********
```

```
for (count = 0; count < size; count++) {
     for (int i = 0; i < size - count; i++)
        System.out.print(" ");
     for (int k = 0; k <= count; k++)
        System.out.print("*");

     System.out.println();
}
```

(b) Write a block of code to print the original triangle flipped ***horizontally*** with an output of:

```
**********
*********
********
*******
******
*****
****
***
**
*
```

```
for (count = 0; count < size; count++) {
     for (int i = 0; i < size - count; i++)
        System.out.print("*");
     for (int k = 0; k < count; k++)
        System.out.print(" ");

     System.out.println();

}
```

(c) Write a block of code to print the original triangle flipped both ***vertically*** and ***horizontally*** with an output of:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

```
for (count = 0; count < size; count++) {
     for (int i = 0; i <= count; i++)
        System.out.print("*");
     for (int k = 0; k < size - count; k++)
        System.out.print(" ");

      System.out.println();
}
```

Marking Guide

+ 2 marks for each loop in each of parts (a), (b), (c)
+ For (b) and (c) the trailing blank spaces need not be coded

Part III - Program writing - 35 Marks

9. Study the following, mostly uncommented code for a simple class to represent parts. Your answers to various parts of this question must appear in the spaces provided in subsequent pages.

```
class Part
{
   private String ID;      // part ID (ID)
   private String name;    // part name (name)
   private double uPrice; // part unit price (uPrice)
   private int stLevel;    // part stock level (stLevel)
```

// Part (a) (3 marks). Write the declaration of an instance variable called reLevel to represent // the reorder level.

```
private int reLevel; // 1 mark each for the three components
```

// Part (b) (7 marks). Write a constructor that receives 4 paramaters representing a part name, // part ID, unit price and stock level,and initializes corresponding instance variables. The reorder // level should be set to the same value as the stock level

```
public Part(String pn, String pid, double up, int sl)
{   // 2 marks for header + parameters
    name = pn; // 1 mark for each of the remaining initialisations
    ID = pid;
    uPrice = up;
    stLevel = sl;
    reLevel = sl;
}

   public String getID(){return ID;}
   public String getName(){return name;}
   public double getUPrice(){return uPrice;}
   public double getSLevel(){return stLevel;}
   public void replenish(int qty){stLevel += qty;}

   public double supply(int qty) // Supply qty units of part
   {
      if ( qty > getSLevel() )  // check enough stock
          return -1.0;
      stLevel -= qty;
      if (getSLevel() < getRLevel() ) // Stock lvl < reorder lvl
         System.out.println("Reorder Part " + ID + ");
      double price = qty * getUprice(); // Compute purchase price
      return price;
   }
```

// Part (d) (10 marks). In the space provided below: // Should have been part (c) and next one (d)
// (i) Write a getter/accessor method called getRLevel() to return the reorder level
// (ii) Write a setter/mutator method called reduce(int q) to reduce the stock level by the amount in // the parameter q, with the appropriate return type chosen for the method

(i)

```
public int getRLevel() { return reLevel; }

// 3 marks header; 2 marks body
```

(ii)

```
public void reduce(int q) { stLevel -= q; }

// 3 marks header; 2 marks body
```

```
   public void print() // Print properties of this part
   {
      System.out.println("ID = " + ID);
      System.out.println("name = " + name);
      System.out.println("unit Price = " + uPrice);
      System.out.println("Stock Level = " + stLevel);
   }
} // End of Part class
```

// Part (e) (15 marks). For this part of the question you may find it easier to detach the last page,
// and then answer the questions below in the space provided thereafter.
//
// (i) Write a statement to add a new part to the array, with details "P128", "Wheel", 300, 111
// and update any other relevant code, if any.
// (ii) Write code to print details of all parts currently held in the array
// (iii) State clearly what is returned by the method searchPart if partID is "P124"

(i) // 5 marks (3 marks for use of new and assignment; 2 for updating pc)

```
p[5] = new Part("P128","Wheel",300.00,111);
      pc = 6; // There are now 6 Part objects referenced by the
              // array, in p[0] to p[pc - 1].
```

(ii) // 5 marks (3 for loop and 2 for body; if print() is not used then deduct a mark)

```
for (int k = 0; k < pc; k++)

    p[k].print()
```

(iii) // 5 marks (3 marks for mention of specific Part object in array containing "P128", and therefore the reference returned to this element in the array)

```
searchPart() returns a reference to the Part object containing ID
"P128", since this part appears in one of the Part instances in the
array p
```

```
public class TestParts2
{
   public static void main(String args[])
   {
      Part p[] = new Part[50]; // Up to 50 parts may be stored
      int pc; // Counter to tell us how many parts in array

      p[0] = new Part("P123","Axle",120.00,78);
      p[1] = new Part("P124","Shock Absorber",180.00,32);
      p[2] = new Part("P125","Brake",72.50,325);
      p[3] = new Part("P126","Exhaust Pipe",155.50,170);
      p[4] = new Part("P127","Front Panel",310.00,124);
      pc = 5; // There are now 5 Part objects referenced by the
              // array, in p[0] to p[pc - 1].

      Scanner sc = new Scanner(System.in);
      do {

          System.out.print("Enter part ID : ");
          String partID = sc.nextLine();
          Part pref;

          pref = searchPart(partID, p, pc);
          if (pref != null)
          {
             System.out.print("Enter qty required : ");
             int qty = sc.nextInt();
             sc.nextLine();
             double cost = pref.supply(qty);
             if (cost < 0 )
                System.out.print("Insuff stock ");
              else System.out.print(pref.getID()+ "  " + qty + "
items = "
                                           + cost);
          }
          else
              System.out.println("No such part");
          System.out.print("Continue Y/N : ");
      } while ( sc.nextLine().charAt(0) == 'Y');
   }


   public static Part searchPart(String ID, Part pt[], int count)
   {
       for (int i = 0; i < count; i++)
       {
           if ( pt[i].getID().compareTo(ID) == 0)
          return pt[i]; // if part with matching ID is found
                        // return reference to this Part instance
       }
       return null; // No matching Part instance, so return null
   }

}
```