SIM UNIVERSITY

ICT131

Examination – January Semester 2015

Introductory Programming and Object Oriented Concepts Using Java

Tuesday, 12 May 2015 **1:00 pm – 3:00 pm**

Time allowed: 2 hours

INSTRUCTIONS TO STUDENTS:

1. This examination contains **FOUR (4)** questions and comprises **ELEVEN (11)** printed pages (including cover page and appendix A).
2. You must answer **ALL** questions.
3. This is a Closed Book examination.
4. All answers must be written in the answer book.
5. Appendix A contains some Java API which you might need.

At the end of the examination
Please ensure that you have written your examination number on each answer book used.

Failure to do so will mean that your work cannot be identified.

If you have used more than one answer book, please tie them together with the string provided.

THE UNIVERSITY RESERVES THE RIGHT NOT TO MARK YOUR SCRIPT IF YOU FAIL TO FOLLOW THESE INSTRUCTIONS.

Answer all the questions. (Total 100 marks)

Question 1

(a) The shaded area enclosed between the arc and the chord, as shown in Figure Q1(a)(i), is a segment.

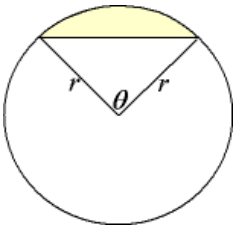

Figure Q1(a)(i)

The formula to calculate the area of a segment is

$$Area = \frac{1}{2}r^2\left(\frac{\theta\pi}{180} - \sin\frac{\theta\pi}{180}\right)$$

where θ is the angle of the arc (in degrees), π is $\frac{22}{7}$ and r is the radius of the circle.

A program skeleton in Figure Q1(a)(ii) calculates and displays the area of the segment when the radius and angle are entered as the command line argument.

```
 1 public class Q1a
 2 {
 3    public static void main (String [] args)
 4    {
 5     (i) // declare r, the radius whose value is args[0]
 6    (ii) // declare a, the angle whose value is args[1]
 7
 8   (iii) double area = 1/2 * Math.pow(r, 2) *((a* 22/7*180)
 9          - Math.sin((a* 22/7*180)));
10
11    (iv) System.out.println("Area is " + area +
12          " when radius is " + r + " and angle is " + a);
13    }
14 }
```

Figure Q1(a)(ii)

(i) The radius of a circle, `r` is a real number. Write a Java statement for line 5 that declares and initialises `r`, the radius.

(2 marks)

(ii) The angle of a circle, `a` is a whole number. Write a Java statement for line 6 that declares and initialises `a`, the angle.

(2 marks)

(iii) The Java statement in lines 8 and 9 does not calculate the area correctly. Correct the Java statement as **ONE** statement to reflect the given formula.

(4 marks)

(iv) Rewrite the Java statement in lines 11 and 12 as **ONE** statement, to display the area and radius, both as numbers with 2 decimal places, and the angle as a whole number.

(5 marks)

(b) Examine the program in Figure Q1(b) and outline the output of the program.

```
 1 public class Q1b
 2 {
 3    public static void main (String [] args)
 4    {
 5       int [] c = { 1, 2, 3, 5, 7};
 6       int a = c[1], b = c[4];
 7
 8       System.out.println("(a) " + "c[1]= " + c[1]+
 9          " c[4]= " + c[4]);
10       m1(a , b);
11       m1(b , a);
12       System.out.println("(b) " + "c[1]= " + c[1]+
13          " c[4]= " + c[4]);
14    }
15
16    public static void m1( int a, int b)
17    {
18       System.out.println("(c) " + "a= " + a + " b= " + b +
19          " a%b= " + (a%b));
20       a -= ++b + 1;
21       System.out.println("(d) " + "a= " + a + " b= " + b +
22          " a/b= " + (a/b));
23    }
24 }
```

Figure Q1(b)

(12 marks)

Question 2

(a) The algorithm to convert a decimal number to octal (base 8) number is given in Figure Q2(a)(i).

1. Let *n* be the decimal number.
2. Let *m* be a string, initially empty, the result of the conversion. The composition of the result starts from the right to left.
3. Repeat until *n* becomes 0
   Divide *n* by *8*, letting the result be *d* and the remainder be *r*.
   Append *r* to the left of *m*.
   Let *d* be the new value of *n*.

Figure Q2(a)(i)

Examine the program given in Figure Q2(a)(ii) below, which is an attempt to convert the algorithm in Figure Q2(a)(i) to Java, and answer the questions that follows.

```
 1 import java.util.Scanner;
 2 public class Q2b2
 3 {
 4    public static void main(String args[])
 5    {
 6       Scanner sc = new Scanner( System.in );
 7       System.out.print("Enter a decimal number: ");
 8       int num =sc.nextInt();
 9       int rem;
10       String str="";
11
12       int num1 = num;
13       while(num>0)
14       {
15          num=num/8;
16          rem=num%8;
17          str += rem;
18       }
19       System.out.println("decimal " + num + " = octal " + str);
20    }
```

Figure Q2(a)(ii)

The program was run several times and sometimes it produced incorrect results. Note that user input is underlined.

Run 1:

```
Enter a decimal number: 7
decimal 0 = octal 0
```

Note that the output should have been: `decimal 7 = octal 7`

Run 2:

```
Enter a decimal number: 8
decimal 0 = octal 10
```

Note that the output is correct!

Run 3:

```
Enter a decimal number: 9
decimal 0 = octal 10
```

Note that the output should have been: `decimal 9 = octal 11`

Run 4:

```
Enter a decimal number: 725
decimal 0 = octal 2310
```

Note that the output should have been: `decimal 725 = octal 1325`

Correct the logic of the program. Your answer must show clearly your changes in **complete** Java statements. Use line numbers in Figure Q2(a)(ii) and

prepositions such as: at, before or after, to specify where your statements are to be added, removed or changed.

(12 marks)

(b) The program in Figure Q2(a)(ii) did not perform any input validation. A valid input value for this program is a non-negative whole number. Outline the control structures of structured programming to perform the validation.

Modify the program by developing the control structures to check for the input to be **read as a string**. Display one or more of the messages when the input is invalid and repeat the input process until a valid input is entered.

A sample run is shown below with user input underlined:

```
Enter a decimal number: -1
Input should not have any negative sign!
Enter a decimal number: -a
Input should not have any negative sign!
Input should not contain letters!
Enter a decimal number: -1.1
Input should not have any decimal point!
Input should not have any negative sign!
Enter a decimal number: -1.a
Input should not have any decimal point!
Input should not have any negative sign!
Input should not contain letters!
Enter a decimal number: 15
decimal 15 = octal 17
```

(13 marks)

Question 3

Develop a program in Java using the control structures of structured programming on an array data structure.

(a) Write Java statements that declare and create three arrays to store the following information for a competition with 10 contestants, their positions and prizes:

- name of contestant: a name consists of letters, including space.
- position: the array should store the contestant position in order of their final rank.
- prize money: initialise the array at declaration time with these values - 5000, 3000, 2000, 1000, 500, 400, 300, 200, 100, 50

(3 marks)

(b) Write a Java method with the following method signature:

```
public static void initContestants(String [] names)
```

This method repeatedly prompts and reads the names of all the contestants and stores it in the corresponding element of the `names` array.

(4 marks)

(c) Write a Java method with the following method signature:

```
public static void remove (int [] numbers, int pos)
```

Refer to Figure Q3(c). This method removes the array element at `pos` by shifting the contents of the array to fill the gap and then setting the last element of the array to 0.

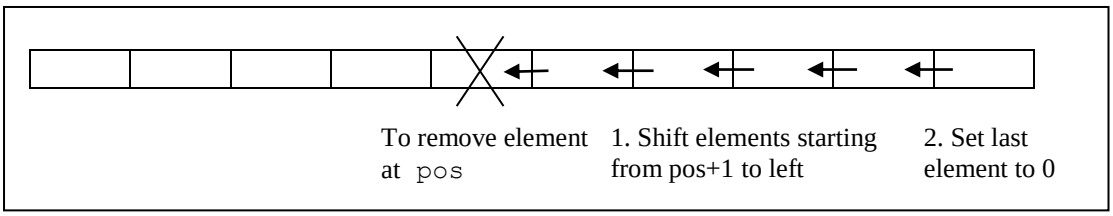

Figure Q3(c)

(5 marks)

(d) Write a Java method with the following method signature:

```
public static void rankContestants(int [] positions)
```

`rankContestants` uses a local array variable named `contestants`. This local array initially stores 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.

Refer to Figure Q3(d)(i). To fill the `positions` array, the method generates a random number from 0 to a maximum of 9 to pick the value stored in the `contestants` array at the element as specified by the random number. For example, if the random number generated is 4, then the fifth element will be removed from the `contestants` array and the value of the fifth element placed into the `positions` array. The picked element is removed using the method in Question 3(c).

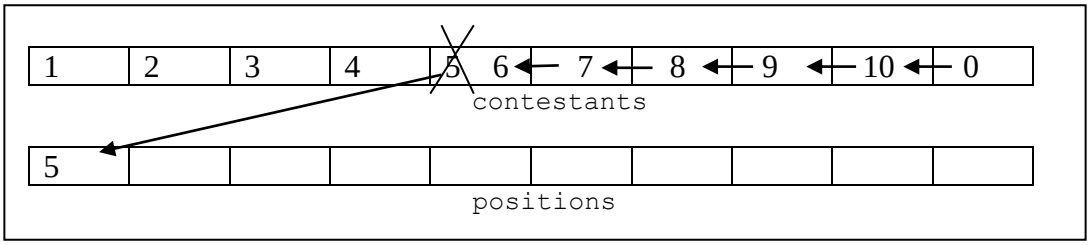

Figure Q3(d)(i)

The process then repeats, this time, a random number from 0 to 8, or 1 less than the previous maximum is generated. Suppose now, the random number generated is 1. Then the second iteration will result in Figure Q3(d)(ii).

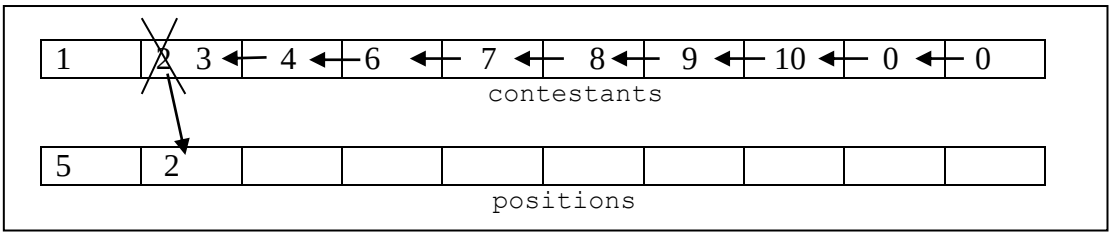

Figure Q3(d)(ii)

The process repeats until the `positions` array is filled and the `contestants` array contains only zeroes.

(13 marks)

Question 4

A company producing talent shows would like to computerise the management of its contestants. You are engaged to implement the classes required for this object-oriented application and to develop a prototype using the object-oriented programming approach. Each contestant has a name, age, gender (M or F), score and whether the contestant is currently employed in the entertainment industry.

You are asked to implement the `Contestant` class for the object-oriented application.

(a) Write a Java class that will model the contestants in the talent show. This class should have the following:

- Suitable instance variables for each of the information given above.
- A constructor with the appropriate actions on the instance variables.
- The usual get and set methods for all the instance variables.
- A `toString()` method that returns the values of the instance variables with descriptions.

(15 marks)

(b) Write a separate test program using object oriented programming approach to exercise the class written in Question 4(a). The test program should perform the following tasks:

(i) Create 4 contestants as shown in the table below and hold them in a single suitable data structure:

Name	Age	Gender	Score	Employed in entertainment industry?
Joleen Cai	18	F	7.3	No
Mark de Souza	19	M	7.9	No
Yunos Sahid	24	M	8.1	Yes
Saina Jayanthi	22	F	8.0	No

Table Q4(b)

(ii) Update scores for only contestants that

- are in the entertainment industry with score below 8.5 by decreasing the score by 0.5 and
- are not in the entertainment industry with score of 8 or above by increasing the score by 0.5

Display the contestants together with the score **before** and **after** being deducted or added, whenever the scores are changed.

A sample output is given here:

```
Name: Yunos Sahid Age: 24 Gender: M Score: 8.1 Employed in
Entertainment industry: true -- Deduct 0.5 = 7.6
Name: Saina Jayanthi Age: 22 Gender: F Score: 8.0 Employed
in Entertainment industry: false -- Add 0.5 = 8.5
```

(10 marks)

Appendix A

Class Double

Modifier and Type	Method and Description
`double`	`doubleValue()` Returns the `double` value of this `Double` object.
`int`	`intValue()` Returns the value of this `Double` as an `int` (by casting to type `int`).
`static double`	`parseDouble(String s)` Returns a new `double` initialized to the value represented by the specified `String`, as performed by the `valueOf` method of class `Double`.

Class Integer

Modifier and Type	Method and Description
`double`	`doubleValue()` Returns the value of this `Integer` as a `double`.
`int`	`intValue()` Returns the value of this `Integer` as an `int`.
`static int`	`parseInt(String s)` Parses the string argument as a signed decimal integer.
`static Integer`	`valueOf(int i)` Returns an `Integer` instance representing the specified `int` value.

Class Math

Modifier and Type	Method and Description
`static double`	`ceil(double a)` Returns the smallest (closest to negative infinity) `double` value that is greater than or equal to the argument and is equal to a mathematical integer.
`static double`	`floor(double a)` Returns the largest (closest to positive infinity) `double` value that is less than or equal to the argument and is equal to a mathematical integer.

`static double`	`pow(double a, double b)` Returns the value of the first argument raised to the power of the second argument.
`static double`	`random()` Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.
`static double`	`sin(double a)` `Returns the trigonometric sine of an angle.`

Class Scanner

Modifier and Type	Method and Description
`boolean`	`hasNextBoolean()` Returns true if the next token in this scanner's input can be interpreted as a boolean value using a case insensitive pattern created from the string "true|false".
`boolean`	`hasNextDouble()` Returns true if the next token in this scanner's input can be interpreted as a double value using the `nextDouble()` method.
`boolean`	`hasNextInt()` Returns true if the next token in this scanner's input can be interpreted as an int value in the default radix using the `nextInt()` method.
`String`	`next()` Finds and returns the next complete token from this scanner.
`boolean`	`nextBoolean()` Scans the next token of the input into a boolean value and returns that value.
`double`	`nextDouble()` Scans the next token of the input as a`double`.
`int`	`nextInt()` Scans the next token of the input as an `int`.
`String`	`nextLine()` Advances this scanner past the current line and returns the input that was skipped.

Class String

Modifier and Type	Method and Description
char	charAt(int index) Returns the char value at the specified index.
int	compareTo(String anotherString) Compares two strings lexicographically.
int	compareToIgnoreCase(String str) Compares two strings lexicographically, ignoring case differences.
String	concat(String str) Concatenates the specified string to the end of this string.
int	indexOf(String str) Returns the index within this string of the first occurrence of the specified substring.
int	indexOf(String str, int fromIndex) Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
int	lastIndexOf(String str) Returns the index within this string of the last occurrence of the specified substring
int	length() Returns the length of this string.

----- END OF PAPER -----