بسم الله الرحمن الرحيم

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

Information and Computer Science Department

2011/2012 Third Semester (Term 103)

ICS102: Introduction to Computing (2-3-3)

FINAL EXAM

Wednesday, August 17th 2011, 09:00 AM – 11:00 AM

120 MINUTES

Student Information

Name:	**Key Solution**
ID:	
Section:	

Question No.	Maximum Score	Score
01	30	
02	16	
03	16	
04	8	
05	30	
TOTAL	100	

```
import java.LuckStream;
System.out.println(LuckStream.GoodLuck);
```

Question 1 (30 points):

Find the output of the following Java code snippets:

I)

```
int [][] a = {{4,9,2,12},{16,35,18,21},{8,6,15,22}};
int i, j;
for(i = 0 ; i < a.length ; i++) {
    for(j = 0; j < a[i].length ; j++) {
        if (a[i][j] % 3 != 0)
            continue;
        System.out.print(a[i][j] + "\t");
        if (i % 2 != 0)
            break;
    }
    System.out.println();
}
```

OUTPUT

```
9    12
18
6    15
```

II)

```
Scanner kb = new Scanner(System.in);
int [][] b = new int [3][3];
int [] a = new int[b.length * 2];
for(int j = 0; j < b[0].length; j++){
    for(int i = 0; i < b.length ; i++)
       b[i][j] = kb.nextInt();
}
for(int j = 1 ; j <= 2; j++) {
    for(int i = 0; i <= 2; i += 2)
      System.out.print(b[i][j] + "\t");
}
System.out.println();
for(int i = 0; i < 6; i++) {
    a[i] = b[i%2][i%3];
    System.out.print(a[i] + "\t");
}
```

USER INPUT

```
11   2   5
3    6   8
7    9   14
```

OUTPUT

```
3    8    7    14
11   6    7    2    3    9
```

Question 2 (16 points):

Find the output of the following Java program:

```
class Test{
  public static void changeLocation(String loc){
     loc = "4-100";
  }

  public static void changeCredit(double credit){
     credit = 4.0;
  }

  public static int incrementSection(int section){
     section++;
     return section;
  }

  public static void changeCourse(String course, int code,
                                  double credit, String [] locations,
                                  int [] sections){
    course = "MATH";
    code = 101;
    changeCredit(credit);
    System.out.println(course + "\t" + code + "\t" + credit);
    locations[0] = "5-400";
    changeLocation(locations[1]);
    sections = new int[2];
    sections[1] = 52;
    System.out.println(sections[0]);
  }

 public static void main(String[] args){
  String course = "ICS";
  int code = 102;
  double credit = 3.0;
  String [] locations = {"22-130" , "23-18"};
  int [] sections = {1 , 2};
  changeCourse(course,code,credit,locations,sections);
  System.out.println(locations[0] + "\t" + locations[1]);
  sections[0] = incrementSection(sections[1]);
  System.out.println(sections[0] + "\t" + sections[1]);
 }
}
```

OUTPUT

```
MATH    101    3.0
0
5-400       23-18
3     2
```

Question 3 (16 points):

Find the output of the following Java program:

```
public static void exchange(int[] s1, int [] s2, int n){
  int i, t;
  for (i = 0; s1[i] != 1 && s2[i] != 5; i++) {
    t = s1[i];
    s1[i] = s2[n-1-i];
    s2[n-1-i] = t;
  }
}

public static void main(String[] args) {
  int [] m = {5,6,7,8};
  int [] n = {1,2,3,4};
  exchange(m , n ,4);
  for(int i = 0; i < 4 ; i++)
    System.out.println(m[i] + "\t" + n[i]);
}
```

OUTPUT

```
4    1
3    7
2    6
8    5
```

Question 4: (8 points)

Write a public static **method** isRagged **that takes an** int **2D array** data **as an argument, and returns** true **as soon as it discovers that** data **is a ragged array. If** data **is found to be not ragged, then** isRagged **should return** false**. If** data **is not referencing an existing object,** isRagged **should print an error message then stops the program that called this method.**

```
public static boolean isRagged(int [][] data){

    if (data == null){

        System.out.println("Error, data is null");

        System.exit(0);

    }

    for (int [] i : data)

        if (i.length != data[0].length) return true;

    return false;

}
```

Question 5: (30 points)

Complete the implementation of the following class Student **as denoted by the enclosed comments:**

```
public class Student{
/* Define instance variables name (String) , id (long) and scores (array
/* of double). These instance variables must NOT be accessable from code
/* outside the class Student */
```

```
private String name;

private long id;

private double [] scores;
```

```
/* Define the constant INSTITUTION (String) and set it to "KFUPM". This
 * constant can be accessed from code outside class Student, and also
 * could be accessed without instantiating an object of class Student */
```

```
public static final String INSTITUTION = "KFUPM";
```

```
/* Define a copy constructor that takes an object of class Student as an
 *  argument, then uses it to initialize instance variables. This
 * constructor should be accessable from code outside class Student */
```

```
public Student(Student student)

{

    this.name = student.name;

    this.id = student.id;

    this.scores = student.scores.clone();

}
```

```
/*  Define an accessor method for name that could be used from code
 *  outside class Student, this method must be called by instantiated
 * objects only */
```

```
public String getName()

{ return name; }
```

```
/*  Define a mutator method for id that could be used from code inside
 *  class Student only, this method must be called by instantiated
 *  objects only, and must NOT accept negative values for id */
private void setId(long id)
{
    if (id >= 0) this.id = id;
}

/*  Define toString method that could be used from code outside
 *  class Student, this method returns the values of the instant
 *  variables name and id (separated by a tab) as a String object.
 *  This method must be called by instantiated objects only */
public String toString()
{
    return name + " " + id;
}

/*  Define average method that could be used from code outside
 *  class Student, this method returns the average value of the
 *  array scores. This method must be called by instantiated
 *  objects only */
public double average()
{
    double sum = 0;
    for (int score : scores)
        sum += score;
    return sum/scores.length;
}
```

```
/*  Define equals method that could be used from code outside
 *  class Student, this method takes an object of class Student as
 *  an argument and returns true if the contents of the instance
 *  variables are identical with the current object, otherwise it
 *  returns false. This method must be called by instantiated
 *  objects only */
public boolean equals(Student student)
{
    if (!this.name.equals(student.name))
        return false;
    if (this.id != student.id)
        return false;
    if (this.scores.length != student.scores.length)
        return false;
    for (int i = 0; i < this.scores.length; i++)
        if (this.scores[i] != student.scores[i])
            return false;
    return true;
}
```

Note: Above class is neither complete nor useful as defined above, this is because we wanted to test your understanding according to the limited time and space of this exam and we couldn't include everything. For example, there should be a mutator for every private instance variable or at least have a constructor that sets the private values of these instance variables. However, you ARE NOT REQUIRED to write these missing methods/constructors.

For methods that accept a Student object, you can assume that the passed object is always properly instantiated and doesn't equal to null. Again, this is not a safe assumption that should be adapted always, but we have it here to reduce space needed for this exam.