INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of FIVE (5) questions and comprises TEN (10) printed pages.

2. This is an OPEN BOOK examination.

3. Answer all questions. You may use pencil to write your codes. Pen is preferred for other questions.

4. Fill in your Matriculation Number on the cover page of the Answer Book.

5. Do NOT fill in the question numbers on the cover page of the Answer Book.

6. Do not tie your Answer Book if you use only one Answer Book.

7. If you use more than one Answer Book, tie them together with the strings provided.

8. Do not submit any unused Answer Book.
1. **Dice** [15 marks]

The following program **Dice.java** simulates 6000 rolls of a die to show that each face of the die shows up with approximately equal probability. That is, the output will show that each face (1 through 6) of the die appears approximately 1000 times. The program uses an integer array `freq` to store the frequency of each face of a die. Seven elements are declared for the array, but we use only `freq[1]` to `freq[6]` for the six faces and leave `freq[0]` unused.

The program contains errors (syntax errors, logic errors and omissions). Study the program and identify all the errors. You should indicate the line number where the error occurs and provide the corrected version. For omission, you should indicate what the omission is and where it should appear.

```java
public class dice {
  int[] freq;
  public Dice() {
    freq = new int[7]; // freq[0] is not used
  }
  public void getFreq(int face) {
    return freq[face];
  }
  public void setFreq(int face, int value) {
    freq[face] = value;
  }
  public static void main(String[] args) {
    Dice die = new Dice();
    initFreq();
    printFreq();
  }
  public void initFreq() {
    Random num = new Random();
    int face = 0;
    for (int roll = 1; roll == 6000; roll++) {
      int face = 1 + num.nextInt() * 6;
      setFreq(face, getFreq(face));
    }
  }
  public void printFreq() {
    for (int i = 1; i <= 6; ++i) {
      System.out.println ("Face " + i + " has frequency " + getFreq(face));
    }
  }
}
```
2. **Output** [15 marks]

Study the following classes carefully, and give the output of class `useS`.

```java
class S {
    private static int v1 = 0;
    private int v2, v3;

    public S () {
        this (0, 0);
    }

    public S (int v1) {
        v2 = v1;
        this.v2 = v1;
        this.v3 = v2;
    }

    public S (int v1, int v2) {
        this.v2 = v1;
        this.v3 = v2;
    }

    public void m1 (int v) {
        v1 += v;
    }

    public void m2 (int v) {
        v2 += v;
        v3 += v2;
    }

    public int m3 (int v) {
        int v3 = v;
        v3 += v1;
        return v3;
    }

    public int getv1 () {
        return v1;
    }

    public int getv2 () {
        return v2;
    }

    public int getv3 () {
        return v3;
    }
}
```
public class useS {
    public static void main(String[] args) {
        S s1 = new S();
        S s2 = new S(5);
        S s3 = new S(5,6);

        s1.m1(5);
        s1.m2(5);
        int x1 = s1.m3(5);
        System.out.println(s1.getv1());
        System.out.println(s1.getv2());
        System.out.println(s1.getv3());
        System.out.println(x1);
        System.out.println();

        s2.m1(6);
        s2.m2(6);
        int x2 = s2.m3(6);
        System.out.println(s2.getv1());
        System.out.println(s2.getv2());
        System.out.println(s2.getv3());
        System.out.println(x2);
        System.out.println();

        s3.m1(7);
        s3.m2(7);
        int x3 = s3.m3(7);
        System.out.println(s3.getv1());
        System.out.println(s3.getv2());
        System.out.println(s3.getv3());
        System.out.println(x3);
        System.out.println();

        System.out.println(x1 + x2 + x3);
    }
}
3. **Sales** [20 marks]

A company has four salespersons (1 to 4) who sell five products (1 to 5). Every day, each salesperson passes in a sales-slip for each type of product sold. Each sales-slip contains:

i) The salesperson number;
ii) The product number;
iii) The total dollar value of that product sold that day.

Thus, each salesperson passes in between 0 and 5 sales-slip per day. The company needs a program to read all the information for last month’s sales, and summarize the total sales by salesperson by product. All totals should be stored in the two-dimensional array `sales`. After processing all the information for the last month, you are to print the results in a tabular format.

The following code fragment is given. You are not to change anything in the given code.

```java
import java.io.*;
import java.util.StringTokenizer;
import java.text.*;

public class Sales {
    public static void main(String[] args) throws IOException {
        double[][] sales = new double[4][5];
        readSalesSlips(sales);
        printSales(sales);
    }
}
```

a. Write the `readSalesSlips` method to read in all the sales-slips for the last month.

A sample run is shown below. The first line, for example, shows the salesperson 2 who sold product 3 with value $60.50. You may use the StringTokenizer or other method to extract multiple values on a single line of input.

```
Read sales-slips info:
2 3 60.50
4 1 82.90
1 4 13.40
3 2 22.60
1 3 77.70
2 5 8.60
3 3 12.30
<enter>
```
(Question 3 continued…)  

b. Write the `printSales` method to print the total sales by salesperson by product in the `sales` array in a tabular format with each of the rows representing a particular salesperson and each of the columns representing a particular product. Cross total each row to get the total sales by each salesperson, and cross total each column to get the total sales of each product. Your tabular printout should include cross totals to the right of the totalled rows and to the bottom of the totalled columns (see sample output below).

All amounts must be printed in two decimal places.

A sample output, based on the input in part (a), is shown below.

```
0.00  0.00  77.70  13.40  0.00  91.10  
0.00  0.00  60.50  0.00  8.60  69.10  
0.00  22.60  12.30  0.00  0.00  34.90  
82.90  0.00  0.00  0.00  0.00  82.90  
     82.90  22.60  150.50  13.40  8.60
```
4. [25 marks]

a. **Stable Sort** [5 marks]
   A stable sort preserves the original relative position of keys with the same value after sorting. Describe a situation, with an appropriate example, where a stable sort is useful.

b. **Tracing Euclid’s Algorithm** [5 marks]
   The Euclid’s algorithm computes the greatest common divisor of two non-negative integers (not both zero) \(a\) and \(b\). The code is given below.

   ```c
   int GCD(int a, int b)
   {
       int rem;
       while (b > 0) {
           rem = a % b;
           a = b;
           b = rem;
       }
       return a;
   }
   ```

   Given this call \(GCD(170, 289)\), what does it return? Show the values of \(a\) and \(b\) at the end of each iteration in the ‘while’ loop in the given code above.

c. **Analysis of Euclid’s Algorithm** [5 marks]
   What is the worst-case running time of the code in part (b)? (At this level, you are not required to give an exact analysis of the algorithm. A reasonable estimate with justification will be accepted.)
(Question 4 continued…)

d. **Coin Exchange Part 1** [5 marks]

Given an integer value that represents an amount in cents, the method below returns the fewest number of coins (with denominations $1, 50 cents, 20 cents, 10 cents, 5 cents and 1 cent) with the same total value. For example, if amount is 232, then the method returns 6 (two $1 coins, one 20-cent coin, one 10-cent coin, and two 1-cent coins).

Complete the method below by filling in the body of the ‘while’ loop.

```java
public static int numCoins(int amount) {
    int[] denominations = {100, 50, 20, 10, 5, 1};
    int coins = 0;
    while (amount > 0) {
        // code to be filled in
    }
    return coins;
}
```

- 8 of 10 -

e. **Coin Exchange Part 2** [5 marks]

Suppose a new set of coin denominations are used: $1, 40 cents, 30 cents, 20 cents, 5 cents and 1 cent. Hence the initialisation of the array denominations is changed as follows:

```java
int[] denominations = {100, 40, 30, 20, 5, 1};
```

Suppose the rest of the code remains unchanged, (i) give an example and write out the answer produced by the code and the actual answer to show that the code does not give the correct answer, and (ii) explain why it is so.
5. [25 marks]
  
  a. **Ball** [4 marks]
  
  The following code contains the definition of class **Ball**. A Ball object has the `radius` and `color` attributes.

```java
// Ball
import java.awt.*;

public class Ball {
    private double radius;
    private Color color;

    public Ball() { this(10.0, Color.BLUE); }  
    public Ball(double r, Color c) { setRadius(r); setColor(c); }
    public void setRadius(double r) { radius = r; }
    public void setColor(Color c) { color = c; }
    public double getRadius() { return radius; }
    public Color getColor() { return color; }

    public boolean equals(Object v) {
        // to be filled in
    }
}
```

Complete the `equals(Object v)` method.

  
  b. **BallOn2D** [12 marks]

  Extend the **Ball** class in part (a) to a subclass called **BallOn2D**. A BallOn2D object has attributes `radius`, `color`, and a `centre` which is a point in the 2D-plane with integer x- and y-coordinates. Part of the file BallOn2D.java is given below.

```java
private Point centre;

public Point getCentre() { return centre; }

public void setCentre(Point p) { centre = p; }
```

Complete the file by providing:

i. Two constructors. A constructor **BallOn2D()** that creates an object with the default radius and color, and `centre` at (0,0), and an alternative constructor **BallOn2D(Point p)** that creates an object with the default radius and color, and `centre` at p. (Do not write any additional constructor for part ii below.)

ii. A `main()` method that creates two objects `ball1` and `ball2`. Object `ball1` has all default attribute values, and object `ball2` has radius of 1.2, color green, and `centre` at (2,5).
(Question 5 continued…)

iii. A `toString()` method so that `System.out.println(ball1)` and `System.out.println(ball2)` would give the following output:

```
[Radius = 10.0, Color = (0,0,255), Centre = (0,0)]
[Radius = 1.2, Color = (0,255,0), Centre = (2,5)]
```

c. **Recursion** [7 marks]

A prime number is a positive integer whose only distinct factors are 1 and itself. Write a boolean method `isPrime(int n)` that returns `true` if `n` is prime, or `false` otherwise. This method should simply call another auxiliary method, which you need to write, that employs **recursion**. [Only partial credit will be given if you change the formal parameter list of the `isPrime(int n)` method.]

d. **Opinion** [2 marks]

Answer the questions in not more than 50 words.

```java
if (you are an SoC student) {
    Answer these questions:
    \[ What \text{ are you expecting from the coming CS1102 module and how are you going to prepare for the module? } \]
}
else if (you are a non-SoC student) {
    Answer these questions:
    \[ What \text{ faculty are you from (Science, NG, etc.) and how has CS1101 helped or not helped you in your major? } \]
}
```

--- END OF PAPER ---