CS1020 Data Structures and Algorithms I

ANSWER SHEETS

Answers included!

INSTRUCTIONS TO CANDIDATES

1. This document consists of **EIGHT (8)** printed pages.
2. Fill in your Matriculation Number clearly below and at the top of pages 3 and 5.
3. The last two blank pages (pages 7 and 8) may be used if you need more space to write your answers.

MATRICULATION NO.: | | | | | | | | | |

(Write your Matriculation Number legibly with a pen.)

For examiners' use only		
Question	***Max***	***Marks***
Q1-6	6	
Q7	2	
Q8	6	
Q9	6	
Q10	7	
Q11	13	
Total	**40**	

MCQs [6 marks]

Q1.	B	**Q2.**	E	**Q3.**	B
Q4.	A	**Q5.**	D	**Q6.**	C

Q7. [2 marks]

a.	`40.0`
b.	`60.0`

Q8. Output of TestS: [6 marks]

```
10, 6, 8, 12
```

Q9. Game.java [6 marks]

```
public class Game {

   private static final int LIMIT = 1000;

   public static void main(String[] args) {
      Die die = new Die();

      int[] freq = new int[6];

      for (int i=0; i<LIMIT; i++) {
         freq[die.toss() - 1]++;
      }

      int appearedMost = 1;
      int max = freq[appearedMost-1];
      for (int face=2; face<=6; face++) {
         if (freq[face-1] > max) {
            max = freq[face-1];
            appearedMost = face-1;
         }
      }


      System.out.println(appearedMost + 1 +
           " appeared most frequently, a total of "
           + max + " times.");









   }
}
```

Q10.

(a) Modify the constructor. [5 marks]

```
public MyRect(Point v1, Point v2) {

    int minX = Math.min(v1.x, v2.x);
    int maxX = Math.max(v1.x, v2.x);
    int minY = Math.min(v1.y, v2.y);
    int maxY = Math.max(v1.y, v2.y);

    setVertex1(new Point(minX, minY));
    setVertex2(new Point(maxX, maxY));









}
```

(b) A default constructor to create a rectangle with vertices at (0,0) and (1,1). [1 mark]

```
public MyRect() {

    this(new Point(0,0), new Point(1,1));

}
```

(c) Reason to change the two mutators into private methods. [1 mark]

Prevent the client from changing the vertices such that vertex1 (vertex2) is no longer the south-west (north-east) corner.

Q11.

(a) [6 marks]

```
public static MyRect boundingRect( MyRect[] towns )
                    Or ( ArrayList<MyRect> towns )
{
    int minX, maxX, minY, maxY;

    minX = minY = 1000;
    maxX = maxY = 0;

    for (MyRect town: towns) {
        if (town.getVertex1().x < minX)
            minX = town.getVertex1().x;
        if (town.getVertex2().x > maxX)
            maxX = town.getVertex2().x;
        if (town.getVertex1().y < minY)
            minY = town.getVertex1().y;
        if (town.getVertex2().y > maxY)
            maxY = town.getVertex2().y;
    }

    return new MyRect(new Point(minX,minY),
                      new Point(maxX,maxY));









}
```

Q11.

(b) **Version 1: Using array** [7 marks]

```
public static double
          minDistBtwPair ( MyRect[] towns )
{
  double midX1 = (towns[0].getVertex1().x +
                  towns[0].getVertex2().x)/2.0;
  double midY1 = (towns[0].getVertex1().y +
                  towns[0].getVertex2().y)/2.0;

  double midX2 = (towns[1].getVertex1().x +
                  towns[1].getVertex2().x)/2.0;
  double midY2 = (towns[1].getVertex1().y +
                  towns[1].getVertex2().y)/2.0;

  // Initialise min to the distance between centres of first two towns
  // Alternatively, initialise min to 1414.214 (sqrt(1000^2 + 1000^2))
  double min = Math.hypot(midX1 - midX2, midY1 - midY2);
  double dist;

  for (int i=0; i<towns.length - 1; i++) {
     for (int j=i+1; j<towns.length; j++) {
        midX1 = (towns[i].getVertex1().x +
                 towns[i].getVertex2().x)/2.0;
        midY1 = (towns[i].getVertex1().y +
                 towns[i].getVertex2().y)/2.0;

        midX2 = (towns[j].getVertex1().x +
                 towns[j].getVertex2().x)/2.0;
        midY2 = (towns[j].getVertex1().y +
                 towns[j].getVertex2().y)/2.0;
        dist = Math.hypot(midX1 - midX2, midY1 - midY2);
        if (dist < min) {
           min = dist;
        }
     }
  }
  return min;









}
```

Q11.

(b) **Version 2: Using ArrayList** [7 marks]

```
public static double
          minDistBtwPair ( ArrayList<MyRect> towns )
{
  double midX1 = (towns.get(0).getVertex1().x +
                  towns.get(0).getVertex2().x)/2.0;
  double midY1 = (towns.get(0).getVertex1().y +
                  towns.get(0).getVertex2().y)/2.0;

  double midX2 = (towns.get(1).getVertex1().x +
                  towns.get(1).getVertex2().x)/2.0;
  double midY2 = (towns.get(1).getVertex1().y +
                  towns.get(1).getVertex2().y)/2.0;

  // Initialise min to the distance between centres of first two towns
  // Alternatively, initialise min to 1414.214 (sqrt(1000^2 + 1000^2))
  double min = Math.hypot(midX1 - midX2, midY1 - midY2);
  double dist;

  for (int i=0; i<towns.size() - 1; i++) {
     for (int j=i+1; j<towns.size(); j++) {
        midX1 = (towns.get(i).getVertex1().x +
                 towns.get(i).getVertex2().x)/2.0;
        midY1 = (towns.get(i).getVertex1().y +
                 towns.get(i).getVertex2().y)/2.0;

        midX2 = (towns.get(j).getVertex1().x +
                 towns.get(j).getVertex2().x)/2.0;
        midY2 = (towns.get(j).getVertex1().y +
                 towns.get(j).getVertex2().y)/2.0;
        dist = Math.hypot(midX1 - midX2, midY1 - midY2);
        if (dist < min) {
           min = dist;
        }
     }
  }
  return min;









}
```

This page is intentionally left blank.
Do **NOT** use it for your rough work.
Use it ONLY if you need extra space for your answer, in which case please indicate the **question number clearly**.

This page is intentionally left blank.
Do **NOT** use it for your rough work.
Use it ONLY if you need extra space for your answer, in which case please indicate the **question number clearly**.

— END —