

Dragon Curves in Loop

Test Framework for Median-of-5

Partial Solutions for Exercises 02

1.2.35 *Dragon curves*. Write a program to print the instructions for drawing the dragon curves of order 0 through 5. The instructions are strings of F, L, and R characters, where F means “draw line while moving 1 unit forward,” L means “turn left,” and R means “turn right.” A dragon curve of order n is formed when you fold a strip of paper in half n times, then unfold to right angles. The key to solving this problem is to note that a curve of order n is a curve of order $n-1$ followed by an L followed by a curve of order $n-1$ traversed in reverse order, and then to figure out a similar description for the reverse curve.



Dragon curves of order 0, 1, 2, and 3

```
JavaUI\work\WarmUp03_code>javac DragonCurvesInLoop.java  
H:\course\2018-2019A\JavaUI\work\WarmUp03_code>java DragonCurvesInLoop 5  
0 : F  
1 : FLF  
2 : FLFLFRF  
3 : FLFLFRFLFLFRFRF  
4 : FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRF  
5 : FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFRFRFLFLFRFRFRFLFLFRFRF  
H:\course\2018-2019A\JavaUI\work\WarmUp03_code>
```

```
DragonCurvesInLoop.java X
```

```
1 public class DragonCurvesInLoop {
2     public static void main (String[] args) {
3         int n = Integer.parseInt( args[0] );
4
5         String d = "F", r = "F";
6         for (int i = 0; i <= n; i++) {
7             System.out.printf( "%d : %s\n", i, d );
8             String d_ = d, r_ = r;
9             d = d_ + "L" + r_;
10            r = d_ + "R" + r_;
11        }
12    }
13 }
```

To be
Drawn
...

```
H:\course\2018-2019A\JavaUI\work\WarmUp03_code>javac DragonCurvesInLoop.java

H:\course\2018-2019A\JavaUI\work\WarmUp03_code>java DragonCurvesInLoop 5
0 : F
1 : FLF
2 : FLFLFRF
3 : FLFLFRFLFLFRFRF
4 : FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRF
5 : FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRF

H:\course\2018-2019A\JavaUI\work\WarmUp03_code>
```

1.3.43 *Median-of-5.* Write a program that takes five distinct integers as command-line arguments and prints the median value (the value such that two of the other integers are smaller and two are larger). *Extra credit:* Solve the problem with a program that compares values fewer than 7 times for any given input.

Just Give a Test Framework for Median-of-5 Problem.

The Reference Solution will be delivered next week.

Test Framework for Median-of-5 Problem:

```
1 public class TestMedianOf5Ints {
2     public static void main (String[] args) {
3         for (int i = 1; i <= 5; i++)
4             for (int j = 1; j <= 5; j++)
5                 for (int k = 1; k <= 5; k++)
6                     for (int m = 1; m <= 5; m++)
7                         for (int n = 1; n <= 5; n++)
8                             if (i!=j && j!=k && k!=m && m!=n) { // permutation of 1..5
9                                 int a=i, b=j, c=k, d=m, e=n;
10                                // ...
11                                // code to find out median-of-5
12                                // ...
13                                if (c != 3)
14                                    System.out.printf(
15                                        "[%d %d %d %d %d] ==> [%d %d %d %d %d]\n",
16                                        i, j, k, m, n,          a, b, c, d, e
17                                    );
18                            }
19                    }
20            }
21        }
22    }
23 }
24 }
25 }
```

1.3.5 Write a program `RollLoadedDie` that prints the result of rolling a loaded die such that the probability of getting a 1, 2, 3, 4, or 5 is $1/8$ and the probability of getting a 6 is $3/8$.

```
H:\work\JavaProg\2018Spring\WarmUp03>javac RollLoadedDie.java

H:\work\JavaProg\2018Spring\WarmUp03>java RollLoadedDie 10000
[1250, 1212, 1226, 1246, 1280, 3786]
[0.125, 0.1212, 0.1226, 0.1246, 0.128, 0.3786]

H:\work\JavaProg\2018Spring\WarmUp03>java RollLoadedDie 100000
[12430, 12511, 12592, 12414, 12487, 37566]
[0.1243, 0.12511, 0.12592, 0.12414, 0.12487, 0.37566]

H:\work\JavaProg\2018Spring\WarmUp03>java RollLoadedDie 800000
[100270, 100295, 99913, 100109, 99474, 299939]
[0.1253375, 0.12536875, 0.12489125, 0.12513625, 0.1243425, 0.37492375]

H:\work\JavaProg\2018Spring\WarmUp03>java RollLoadedDie 8000000
[997862, 1001309, 998750, 999927, 999938, 3002214]
[0.12473275, 0.125163625, 0.12484375, 0.124990875, 0.12499225, 0.37527675]

H:\work\JavaProg\2018Spring\WarmUp03>java RollLoadedDie 80000000
[10000748, 9997549, 9999273, 10002315, 9999347, 30000768]
[0.12500935, 0.1249693625, 0.1249909125, 0.1250289375, 0.1249918375, 0.3750096]
```

```
1  import java.util.Arrays;
2
3  public class RollLoadedDie {    // RollLoadedDie.java
4      public static void main (String[] args) {
5          int N = Integer.parseInt( args[0] );
6          int[] count = new int[6];
7          for (int t = 0; t < N; t++)
8              count[ rollDie()-1 ]++;
9
10         double[] frequency = new double[6];
11         for (int i = 0; i < 6; i++)
12             frequency[i] = (double)count[i] / N;
13
14         System.out.println( Arrays.toString(count) );
15         System.out.println( Arrays.toString(frequency) );
16     }
17
18     public static int rollDie () {
19         int die = 1 + (int)(Math.random() * 8);
20         if (die > 6) die = 6;
21         return die;
22     }
23 }
```

1.3.8 Rewrite `TenHellos` to make a program `Hellos` that takes the number of lines to print as a command-line argument. You may assume that the argument is less than 1000. Hint: Use `i % 10` and `i % 100` to determine when to use `st`, `nd`, `rd`, or `th` for printing the `i`th Hello.

Program 1.3.2 Your first while loop

```
public class TenHellos
{
    public static void main(String[] args)
    { // Print 10 Hellos.
        System.out.println("1st Hello");
        System.out.println("2nd Hello");
        System.out.println("3rd Hello");
        int i = 4;
        while (i <= 10)
        { // Print the i-th Hello.
            System.out.println(i + "th Hello");
            i = i + 1;
        }
    }
}
```

```
% java TenHellos
1st Hello
2nd Hello
3rd Hello
4th Hello
5th Hello
6th Hello
7th Hello
8th Hello
9th Hello
10th Hello
```



```
H:\work\JavaProg\2018Spring\WarmUp03>javac Hellos.java
```

```
H:\work\JavaProg\2018Spring\WarmUp03>java Hellos 25
```

→ 1st Hello
→ 2nd Hello
→ 3rd Hello
4th Hello
5th Hello
6th Hello
7th Hello
8th Hello
9th Hello
10th Hello
→ 11th Hello
→ 12th Hello
→ 13th Hello
14th Hello
15th Hello
16th Hello
17th Hello
18th Hello
19th Hello
20th Hello
→ 21st Hello
→ 22nd Hello
→ 23rd Hello
24th Hello
25th Hello

Hellos.java X

```
1 public class Hellos { // Hellos.java
2     public static void main (String[] args) {
3         int LIMIT = Integer.parseInt( args[0] );
4
5         for (int i = 1; i <= LIMIT; i++)
6             System.out.println( i + orderPostfix(i) + " Hello" );
7     }
8
9     public static String orderPostfix (int n) {
10         if (n%100 == 11 || n%100 == 12 || n%100 == 13) return "th";
11         if (n%10 == 1) return "st";
12         if (n%10 == 2) return "nd";
13         if (n%10 == 3) return "rd";
14         return "th";
15     }
16 }
```

1.3.12 Write a program FunctionGrowth that prints a table of the values $\log n$, n , $n \log n$, n^2 , n^3 , and 2^n for $n = 16, 32, 64, \dots, 2,048$. Use tabs (`\t` characters) to align columns.

```
H:\work\JavaProg\2018Spring\WarmUp03>javac FunctionGrowth.java
```

```
H:\work\JavaProg\2018Spring\WarmUp03>java FunctionGrowth
```

$\log(n)$	n	$n \times \log(n)$	n^2	n^3	2^n
2.772589	16	44.361420	256	4096	6.553600e+04
3.465736	32	110.903549	1024	32768	4.294967e+09
4.158883	64	266.168517	4096	262144	1.844674e+19
4.852030	128	621.059874	16384	2097152	3.402824e+38
5.545177	256	1419.565426	65536	16777216	1.157921e+77
6.238325	512	3194.022208	262144	134217728	1.340781e+154
6.931472	1024	7097.827129	1048576	1073741824	Infinity
7.624619	2048	15615.219684	4194304	8589934592	Infinity

```

1 public class FunctionGrowth { // FunctionGrowth.java
2     public static void main (String[] args) {
3         System.out.printf( "%10s%10s%20s%10s%20s%20s\n",
4             "log(n)", "n", "n*log(n)", "n^2", "n^3", "2^n"
5         );
6         long n = 16;
7         while (n <= 2048) {
8             System.out.printf( "%10f%10d%20f%10d%20d%20e\n",
9                 Math.log(n), n, n*Math.log(n), n*n, n*n*n, Math.pow(2,n)
10            );
11            n *= 2;
12        }
13    }
14 }

```