In this lab you will learn and practice how to use the \texttt{StdIn} and \texttt{StdOut} libraries for doing some interactive user input as well as formatted output and also use Linux's I/O redirection features. See Section 1.5 of your text.

**Task#1: Review – StdOut**

The StdOut library provides four useful output functions, as show below.


dd

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{print(String s)}</td>
<td>Print \texttt{s} to standard output</td>
</tr>
<tr>
<td>\texttt{println(String s)}</td>
<td>Print \texttt{s} and a newline to standard output</td>
</tr>
<tr>
<td>\texttt{println()}</td>
<td>Print a newline to standard output</td>
</tr>
<tr>
<td>\texttt{printf(String format, ...)}</td>
<td>Print the arguments to standard output, as specified by the format string format</td>
</tr>
</tbody>
</table>

dd

\texttt{StdOut.printf(\texttt{<how to print>, <what to print>});}

\texttt{<how to print>} is a String (called a \textit{format string}) that encodes exactly how the string to be printed will appear. \texttt{<what to print>} is a comma-separated sequence of variables (there may be zero or more). Here are some simple examples:

- \texttt{StdOut.printf(\texttt{"\n"})} // This is equivalent to \texttt{StdOut.println()} or \texttt{System.out.println()}
- \texttt{StdOut.printf(\texttt{"Hello, World!\n"})} // This is equivalent to \texttt{StdOut.println(\texttt{"Hello, World!"})} // or \texttt{System.out.println(\texttt{"Hello, World!"})}
- \texttt{StdOut.printf(\texttt{"%d\n", x})} // where \texttt{x} is an int variable
  // This is equivalent to \texttt{StdOut.println(x)} or \texttt{System.out.println(x)};
- \texttt{StdOut.printf(\texttt{"%6d\n", x})} // Prints the value of \texttt{x} so that it takes exactly 6 spaces (right justified)
- \texttt{StdOut.printf(\texttt{"%-6d\n", x})} // Prints the value of \texttt{x} so that it takes exactly 6 spaces (left justified)
- \texttt{StdOut.printf(\texttt{"%d %d\n", x, Y})} // Prints the values of \texttt{x} and \texttt{y} separated by a SPACE

The following figure describes more details of the \texttt{StdOut.printf()} function:

\texttt{StdOut.printf(\texttt{"%7.5f", Math.PI})}

The codes used in the print specification (for example, \texttt{f} in the figure above, or \texttt{d} in examples above) indicate the type of value to be printed: \texttt{d} (int), \texttt{f} (double), \texttt{s} (String), \texttt{b} (boolean). Before proceeding, please be sure to review the information above.
Task#2: Using StdOut.printf()

Read the program below carefully.

```java
public class Roots {
    // Prints out a table of Square Roots of numbers from 1 to 10
    public static void main(String[] args) {
        int n = 10;

        for (int i=1; i <= n; i++) {
            StdOut.printf("%d. %f\n", i, Math.sqrt(i));
        }
    }
}
```

Write and run the program above. Please ensure that you have typed the program EXACTLY as shown above. Compile and run it. Observe the results.

Next, modify the print statement to the following:

```java
StdOut.printf("%3d. %5.2f\n", i, Math.sqrt(i));
```

Compile and run the program and observe how the table is now formatted, and the values of square roots are nicely rounded to two decimals places. Play with some other format values and see how the formatting changes.

Finally, add commands to the program above to produce a table that looks exactly like the one shown below:

```
#   Square Root
--------------
 1. 1.00
 2. 1.41
 3. 1.73
 4. 2.00
 5. 2.24
 6. 2.45
 7. 2.65
 8. 2.83
 9. 3.00
10. 3.16
--------------
```

Show your resulting program and its output to the instructor.
Task 3: Review – StdIn

The StdIn library provides several useful input functions to perform interactive input of various types of data. The table below summarizes all the functions available in StdIn library:

Review these quickly and focus especially on the first group of functions. We will use some of these next.

Task 4: First, consider the program shown below:

```java
public class Hello {
    // Inputs a name and says Hello
    public static void main(String[] args) {
        String name;
        StdOut.print("Please enter your name: ");
        name = StdIn.readString();

        StdOut.printf("Hello, %s!\n", name);
    } // main()
} // class Hello
```

The program above prompts the user to enter a name (using Std.String()) and then prints out a greeting (using StdOut.printf()). Enter, compile, and run the program above and make sure you understand how the simple interaction is coded.
**Task 5:** Read, study, enter, compile, and run the program below:

```java
public class AddInts {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int sum = 0;
        for (int i = 0; i < n; i++) {
            int value = StdIn.readInt();
            sum += value;
        }
        StdOut.println("Sum is " + sum);
    }
}
```

Next, create a small data file (data.txt) in the same directory as your AddInt.java program. In it, enter the four numbers shown in the example above (144, 233, 377, and 1024) so that one number appears on each line. Once done, run your program using the command:

```
$ java AddInts 4 < data.txt
```

What happened? You just used Linux’s I/O redirection so that your program, instead of taking input from the keyboard, takes it from the file data.txt.

Try create a much longer data file, say with 20 numbers in it. Then run the program:

```
$ java AddInts 20 < data.txt
```

**Task 5:** Write a program RandomInts.java, that generates n random numbers in the range [1..1000] and prints them out on the screen (one number per line). Here is an example use:

```
$ java RandomInts 5
765
211
90
512
67
```

Once done, You can use the two programs above (RandomInts to generate 20 random numbers, and AddInts to add them):

```
$ java RandomInts 20 | java AddInts 20
```

**Task 6:** Modify AddInts to compute the average of all the integers and print it out (only up to 2 decimal places). Test your program using the commands:

```
$ Java AddInts 4
144
233
377
1024
Average is 444.50
```

Once correct, try the following:

```
$ java RandomInts 1000 | java AddInts 1000
```