We've discovered an opportunity to expand your depth of knowledge in operating systems. Are you puzzled? Very clearly — as many people had questions about redirected input for the last assignment. This assignment will help you better understand redirection, random numbers, and arrays.

Your assignment is write TWO programs. They are: one program that generates some number of random numbers within a limit and another program that analyzes the output of the first program.

The first program, named Assignment4a.java, has the following requirements:

- Allow the user to enter the integer number of random numbers to be generated.
- Allow the user to enter the upper limit of the integer numbers generated.
- Generate some number of random numbers up to and including the upper limit.
    - For example, generating 5 numbers with an upper limit of 25 would include the integers from 1 to 25. Depending upon the random number generator's output the results might look like 1, 9, 25, 17, & 8.
    - Write one number per line.

The second program, named Assignment4b.java has the following requirements:

- Allow the user to enter the upper limit of the integer numbers to be read (or *scanned*).
- Collect the random integers from the user until the user enters an integer that is greater than the upper limit selected in the step above or the scanner cannot detect any more data. (There's an example of this in a previous Chapter.)
- Use an array to count the number of times an integer is input. (See Fig 7.8 on pages 254 and 255 for some insight.) For example if there are 43 occurrences of the number 1, the array indexed by 1 would contain the integer value 43, likewise for all the integers from 1 to the upper limit, inclusive.
- Use a method named **printStat** to print out the integer counts for all the numbers from 1 to the upper limit. Pass the array to the method **printStat** as the ONLY parameter to the method.

Both programs can be tested using the Scanner method to collect data from the user.

However, it is REQUIRED that you use command line parameters for the inputs to earn the full grade. (See Section 7.14 on page 283 for more insight. *Pay attention to the parameters' positions.*)

GRADING:

The total possible score for this program is 100 points.

The following point values will be deducted for the reasons stated:

-100 points	The program does not successfully compile from the command line
-90 points	The program does not run from the command line without error or produces no output. The program does not contain an array.
-70 points	The program compiles, runs, and collects no inputs from the user command line or crashes thereafter or produces no output.
-50 points	The program compiles, runs, collects the inputs, and produces no output. The program does NOT contain the specified method, **printStat**.
-25 points	The program compiles, runs, collects the inputs, and produces the incorrect output. Incorrect output includes errors in the calculations, incorrect years of deposit, or errors in the amount deposited. Please note that calculating a deposit for every year instead of the “years to deposit” falls in this category.
-10 points	There are data formatting errors.
no deductions	The program compiles, runs, collects the correct number of inputs, and generates the correct outputs (see the following page for a *minimal* example).

```
$java Assignment4a 15 10

$1
$4
$1
$5                    NOTE: These numbers are generated by the program.
$7
$9
$13
$9
$4
$3
```
(Note that there are 10 numbers from 1 to 15 displayed above. That is 15 is the first parameter passed to the program and 10 is the second.)

```
$java Assignment4b 15

$1
$4
$1
$5
$7
$9
$13
$9
$4
$3
$99 (The number higher than the upper limit terminated the input which generated
    the output shown below. Also note that the first parameter passed to the
    program is the upper limit.)

$ The distribution for the random numbers from 1 to 15 is shown below:

$
         1   2   3   4   5   6   7   8   9   10
 Units   2   0   1   2   1   0   1   0   1    0
   10s   0   0   1   0   0   -   -   -   -    -
$java Assignment4a 1000 50000>javac Assignment4b –u 1000
//The command above takes the output from 4a and redirects it as input to 4b
$java Assignment4a 1000 50000 > bigFileOfIntegers
$java Assignment4b 1000 < bigFileOfIntegers
//And the two commands above redirect the output from 4a into a file, then 4b
gets its input from that same file.
```

Recommendations:

1. Start simple. Get the programs to behave well with user inputs for the limits before tackling the command line parameters.
2. Once the loop structure is working in Assignment4a, add the random number generation. (It is discussed in a couple of examples in Chapter 7.)
3. Use small numbers to check the looping structures and user data inputs/outputs.
4. Setup Assignment4b in a similar fashion, simple user inputs, then command line inputs, redirection, etc. (You might be so good with command line parsing by this point in time, you can just “lift” the code from Assignment4a.)