There are four required problems for this assignment: problems from chapter 6, pages 143-146: 1, 6, 8, and the problem labeled S below. See the hint on the back of this page.

S. Sort the following integers by their absolute value (i.e., Math.abs). For example, if you were to sort only the numbers \([5, -1, 0, 2]\), the correct answer would be \([0, -1, 2, 5]\). These integers were generated by calling the generateNumbers method on the course web page with the arguments max = 100 and n = 800. You may use any tools that you like to solve this problem, including your solution to this week’s lab. Please print your answer in a 20 x 80 grid like this so that it is easy for me to check your solutions.
Tip for question #8: when a method calls another method (or itself recursively), some space must be consumed in remembering the information about the caller. When you evaluate a recursive method by hand, that space is your “TODO list” of problems to get back to. On a computer it is called a “stack frame” for reasons we’ll see later in the semester. In Java, each stack frame requires constant space. These stack frames count towards the total space consumption of a method.

The following problems are optional. Thinking about them may help you prepare for the exam. If you complete them I will mark and give feedback on your answers, but you will not gain or lose any points on this assignment based on your answers.

Consider the following method:

```java
public static String funny (int n) {
    int x = 1 << n;
    long y = 0;  // long is like int, but has 64 bits instead of 32
    String s = "";
    for (int i = 0; i < x; ++i) {
        s = "" + i;
        y = y * 1103515245 + 12345;
    }
    return s + ": " + y;
}
```

A. Method funny returns a String containing two numbers separated by a colon. What is the first of the two numbers in the string that funny(6) returns?

B. What is a simple change that will improve the efficiency of this method by a constant factor?

C. Rewrite method funny so that it does not use FOR or WHILE loops but still performs essentially the same function (you will need a helper method).

D. Derive a tight upper bound on the run time of method funny, as it is written above.

E. Derive a tight lower bound on the run time of method funny, as it is written above.

F. Derive a tight upper bound on the space required by method funny, as it is written above. This is tricky, in part because it requires you to speculate about aspects of the Java language that we have never discussed; your ideas are more important than the answer.