Problem 12.1

Sorting problems come up regularly in every day life, from organizing your course notes to alphabetizing your CD collection. Consider the following method of sorting a pile of papers:

a. Divide the pile of papers into many piles of two papers each.

b. Sort these individual two-paper piles. (This should be pretty easy!)

c. Finally, merge together pairs of sorted piles of the same size, using the merge function we saw on homework 5. Repeat until all of the papers have been merged into one sorted pile.

To get a sense of how this works, consider the case where we’re sorting eight pieces of paper. We divide them into four piles, sort, and then merge repeatedly as follows:

original pile:
-------------
8 2 3 5 1 4 6 7

pile 1:  pile 2:  pile 3:  pile 4:
-------  -------  -------  -------
8 2     3 5     1 4     6 7

pile 1:  pile 2:
-------  -------
2 8     3 5

pile 3:  pile 4:
-------  -------
1 4     6 7

result:
-------
How many times did we touch pieces of paper to divide the pile into two-paper piles? What about when we sorted those piles? How many paper-touches would be required in the worst-case situation? How many times did we touch pieces of paper to do the first merge? The second? Figure out how to express the number of merges needed in terms of the total number of pieces of paper.

Express the worst-case total number of paper-touches required by this sorting method in terms of the number of pieces of paper.

**Problem 12.2**

Given the background information and Scheme code in the online notes for Chapter 12, design a set of rules for a Turing machine that searches for the first three letters of your first name. If those three letters are present in order from left to right (but not necessarily all in a row), the machine should place an exclamation mark after the third letter. Otherwise, the tape should not be changed.

For example:

```scheme
> (load "turing.scm")
> (execute tom '(_ A B B T O A B T O M A A _) 1 1)
(_ a b b t o a b t o m a a _)
(_ a b b t o a b t o m ! a _)
> (execute tom '(_ A B B M T O A B T O A A _) 1 1)
(_ a b b m o a b t o a a _)
(_ a b b m o a b t o a a _)
```

**Problem 12.3**

This question also requires you to use the Turing machine code provided online. Write rules for a Turing machine that determines if the letters “a” and “b” are both present on a tape. If so, it replaces one instance of the letter “a” with the letter “b” and one instance of the letter “b” with the letter “a”. Otherwise, the tape is left unchanged.

For example:
> (load "turing.scm")
> (execute switch '(_ ALLIGATOR _) 1 1)
(_ alligator _)
(_ alligator _)
> (execute switch '(_ BABBLE _) 1 1)
(_ babble _)
(_ b b b a l e _)