CS361 Homework #2
Due Tuesday, September 28th

1. Prove that the exact solution of

$$f(n) = 4f(n/2) + n^2, \quad f(1) = 0$$

is

$$f(n) = n^2 \log_2 n$$

when n is a power of 2. Instead of just plugging the solution in to the recurrence, use the structure of an inductive proof: show that it's true for the base case, and then show that if it's true for $n/2$, it's also true for n.

2. Using the fact that we have to handle every possible permutation, give a lower bound on the number of comparisons we need to sort a list of 4 elements. Is there in fact an algorithm that sorts lists of size 4 with this number of comparisons?

3. Use the exact recurrence for the average number of comparisons done by Quicksort,

$$f(n) = n - 1 + \frac{2}{n} \sum_{\ell=0}^{n-1} f(\ell) \ ,$$

and the base case $f(0) = 0$, to calculate exactly the average number of comparisons Quicksort does for lists of size $n = 2, 3,$ and 4. Then explain these numbers in terms of what might happen when the algorithm runs: where the pivot ends up, what size lists it calls itself on recursively, and so on. (Feel free to use symmetry to shorten your answer; for instance, having the pivot at the left end produces the same running time as if it were at the right end.)

4. Here is an algorithm for the `partition` step of Quicksort due to Hoare:

```
// partition A[first..last] using A[first] as the partition:
// put the smaller elements on the left, the larger on the right,
// and return the final position of the partition
int partition(A,first,last) {
  p = A[first];
  left = first+1;
  right = last;
  while (left <= right) {
    while (A[left] <= p and left < last) left++;
    while (A[right] > p) right--;
    if (left < right) swap A[left] and A[right];
  }
  // now put pivot in the right place
  if (right > first) swap A[first] and A[right];
  return right;
}
```

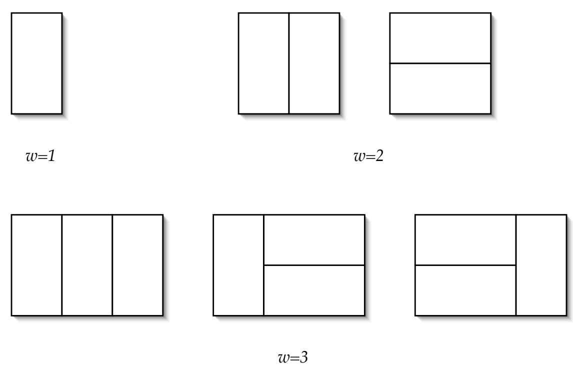
Prove that this algorithm works, by defining a loop invariant which makes some guarantee about the state of the world after each run of the outer `while` loop. Prove "initialization" (base case), "maintenance" (inductive step), and "termination" (making sure that when we're done the entire thing works). If you take more than a page or so, you're probably giving too much detail.

5. In Quicksort, suppose I could guarantee that the pivot's position is a fraction a through the list for some constant $0 < a < 1$. The recurrence for the number of comparisons would then be roughly

$$f(n) = f(an) + f((1-a)n) + n$$

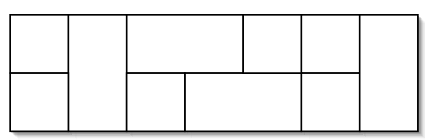
(setting $a = 1/2$ gives the recurrence for Mergesort). Solve this recurrence by plugging in a solution of the form $f(n) = An \log n$. Discuss what happens to the constant A in the limit $a \to 0$ or $a \to 1$.

6. Let $f(w)$ be the number of ways to tile a rectangle of height 2 and width w with horizontal and vertical dominoes. For instance, $f(1) = 1$, $f(2) = 2$, and $f(3) = 3$ as the following picture shows:



What is the recurrence for $f(w)$? What is $f(10)$, for instance? Working backwards, what is $f(0)$?

Extra Credit: What happens if we allow 1×1 blocks as well as dominoes, such as



Hint: it might help to write a recurrence for a *pair* of functions.