In theory, you should know the things on this assignment from previous classes (such as Data Structures and Discrete Math). However, in my experience, very few students remember everything here, and that's OK. Use this assignment as an opportunity to help you figure out some things you should review.

Start early!

Ask for help when you get stuck!

Work together, but remember to write up **your own solutions**.

Have fun!

I **expect** you to ask me for help on homework assignments in this class. If you never ask for any help, you are either very smart or very foolish.

1. How many times do you have to repeatedly halve 32 in order to reach 1? What about 8192? Consider the function which given an input n, outputs the number of times n can be repeatedly halved before falling below 1. What is the common mathematical name for this function?

2. For each of the following, answer with the **best** (smallest) upper bound from this list: $O(1), O(\log n), O(n), O(n \log n), O(n^2), O(2^n)$. Give a **brief justification** for each.

   (a) number of leaves in a depth-n balanced binary tree

   (b) depth of an n-node balanced binary tree

   (c) number of edges in an n-node tree

   (d) time needed to sort a list of n items using merge sort

   (e) number of distinct subsets of a set of n items

   (f) number of bits needed to represent the number n in binary

   (g) time needed to find the closest pair of points among n points in Euclidean space by simply listing all the pairs

   (h) time needed to insert n items into a binary heap

   (i) time needed to find the second largest number in a *sorted* list of n distinct numbers

   (j) time needed to find the second largest number in an *unsorted* list of n distinct numbers

3. Let p_n be defined for all $n \geq 0$ by

   $$p_0 = 0$$
   $$p_n = 2p_{n-1} + 1 \qquad (n > 0)$$

   State and prove a closed formula for p_n.

If you need a refresher on recursion and induction, there are a lot of resources posted on the course website!

4. We will not use the syntax of any particular programming language in this course to specify algorithms. Instead, we will use well-written prose and pseudocode—an intuitive set of instructions that are appropriate to the problem at hand. For example, Algorithm 1 below gives some pseudocode for finding the smallest integer in an array.

Algorithm 1: FINDMIN(A, n)

Require: An array of integers A of length $n \geq 0$. Empty arrays return $+\infty$.

```
1: m ← +∞
2: for i ← 0 to n − 1 do
3:     m ← min(m, A[i])
4: return m
```

Write some pseudocode to find the smallest integer value in a binary tree T with left child $T.left$ and right child $T.right$ and integer value $T.value$. You can suppose that $T.left$ (respectively $T.right$) is NULL if it doesn't have a left (respectively right) child.

Note you should not assume T is a binary *search* tree; that is, there need not be any relationship between the value at a node and the values at its children, so the smallest integer value could be anywhere in the tree.