1. Assume that each word of your machine has 64 bits. Assume that you can multiply two n-word numbers in time $5n^2$ with a standard algorithm. Assume that you can multiply two n-word numbers in time $9n^{\lg 3}$ with a "fancy" algorithm. For each part *briefly justify* and *show your work*.

   (a) Approximately, how large does n have to be for the fancy algorithm to be better?

   (b) How many bits is that?

   (c) How many decimal digits is that?

2. Consider the following recurrence (for the time of some algorithm).

$$T(n) = 5T(n/2) + 2n - 1, \qquad T(1) = 3 \; .$$

   (a) Calculate $T(4)$ by hand. Show your work.

   (b) Use the tree method to solve the recurrence exactly, assuming n is a power of 2. For each subpart *briefly justify* and/or *show your work* when appropriate.

      (i) Draw the tree. You should show at least three levels at the top and at least two levels at the bottom (as done in class).

      (ii) What is the height of the tree? (Note that a tree with one node has height 0, a tree with a root and some children has height 1, etc.)

      (iii) How many leaves are there?

      (iv) What is the total work done by the leaves?

      (v) What is the size of each subproblem at level i? (Note that the root is at level 0, its children are at level 1, etc.)

      (vi) How much work does each subproblem at level i (above the leaves) do?

      (vii) What is the total work for level i (above the leaves)?

      (viii) Write a summation for the total not including the leaves?

      (ix) Simplify the summation.

      (x) What is the total work for the entire algorithm?

3. Bubble Sort can be thought of as a recursive algorithm: Bubble the largest element to the end of the array, and recursively sort the remainder of the elements.

   (a) Write the pseudo-code for this recursive version of Bubble Sort.

   (b) Write a recurrence for the exact number of comparisons.