CMPS 2200 Assignment 2

Name:___________________________________

In this assignment we'll work on applying the methods we've learned to analyze recurrences, and also see their behavior in practice. As with previous assignments, some of of your answers will go in `main.py`.. You should feel free to edit this file with your answers; for handwritten work please scan your work and submit a PDF titled `assignment-02.pdf` and push to your github repository.

1. Derive asymptotic upper bounds for each recurrence below. Here T() means the asymptotic running time. It is work W() when we only have one processor, and will be span S() when we have infinity processors.

- $T(n) = 2T(n/3) + 1$.
  .
  .
  .
  .

- $T(n) = 5T(n/4) + n$.
  .
  .
  .
  .

- $T(n) = 7T(n/7) + n$.
  .
  .
  .
  .

- $T(n) = 9T(n/3) + n^2$.
  .
  .
  .
  .

- $T(n) = 8T(n/2) + n^3$.
  .
  .
  .
  .

- $T(n) = 49T(n/25) + n^{3/2} \log n$.
  .
  .
  .
  .
- $T(n) = T(n-1) + 2$.
  .
  .
  .
  .
- $T(n) = T(n-1) + n^c$, with $c \geq 1$.
  .
  .
  .
  .
- $T(n) = T(\sqrt{n}) + 1$

2. Suppose that for a given task you are choosing between the following three algorithms:

   - Algorithm $\mathcal{A}$ solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
   - Algorithm $\mathcal{B}$ solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.
   - Algorithm $\mathcal{C}$ solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

   What is the recurrence of works for each algorithm? Which algorithm would you choose?

3. Now that you have some practice solving recurrences, let's work on implementing some algorithms. In lecture we discussed a divide and conquer algorithm for integer multiplication. This algorithm takes as input two n-bit strings $x = \langle x_L, x_R \rangle$ and $y = \langle y_L, y_R \rangle$ and computes the product xy by using the fact that $xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$. Write the algorithm specification in SPARC. Then, use the stub functions in `main.py` to implement two algorithms for integer multiplication: a divide and conquer algorithm that runs in quadratic time, and the Karatsaba-Ofman algorithm running in subquadratic time. Then test the empirical

running times across a variety of inputs to test whether your code scales in the manner described by the asymptotic runtime (meaning work here as we implement in sequentially).