Keith Strickling
260674699

Assignment 4

Question 1

My parallelization strategy is as follows: I split up the string into min(length of string, number of threads) sections. Then, I allow each thread to handle a separate section of the string. The string is represented as a linked list, and given that there is only ever one thread acting on a given section of the list at a time, each thread is free to add nodes to the list as it needs. However, because a thread may add nodes to the list and there are multiple production rules being used, we must mark any freshly added nodes so that the next production rule ignores them. Note that I do not provide the most optimal solution. Rather than maintaining pointers into the list throughout all generations, I instead start with a single pointer at the head of the list in each iteration, and must move that pointer to where it needs to be in the list each time. As an optimization, I could count the number of nodes added to a given section during a given generation, and use these numbers to modify pointers into the list for the next iteration.

Question 2

Unfortunately, there is at least one bug in my code that I did not have time to track down before the deadline. However, I believe my strategy is correct and I will briefly explain it here. For parallelization, I use OpenMp sections. The first section has the non-optimistic thread look for matches within the first part of the string. The second section includes a #pragma omp for loop that uses a number of threads equal to the number of optimistic threads, each of which is responsible for finding matchings in a part of the string unique to that thread. For its part of the string, each optimistic thread tries starting from each of the five states and records the results. After the pragma opm sections block is complete, we have all the results and seek to combine them.