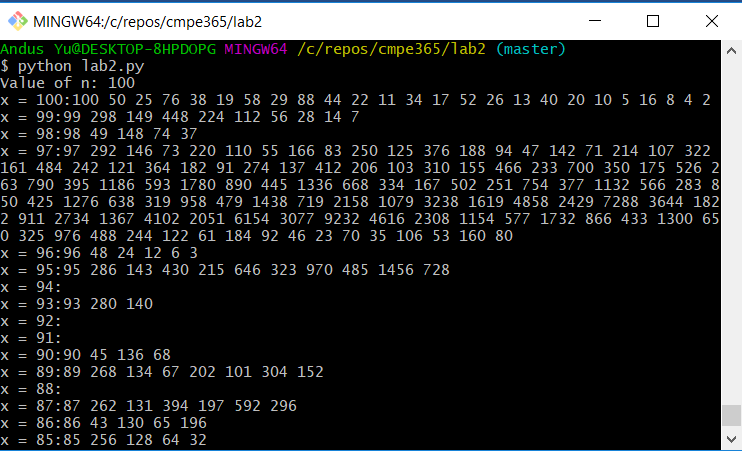
# Lab 2

1. The intermediate values for each value n is recorded onto a text file after each run. Each iteration is printed onto the console in the format below.



Iterations with no output mean that the first term has already been seen somewhere in the intermediate steps (example 76 was seen in iteration 1). After running the algorithm with n = 20,40,60,80,100,200, the maximum intermediate values were found to be 160,9232,9232,9232,9232, and 9232 respectively. With these results, we can conclude that up to a certain n value, the maximum intermediate value will be the same for all of them. For integers greater than that n value, the maximum intermediate value will be increased and will similarly stay the same for another subset of consecutive numbers.

2. This algorithm on average saves a lot of time. Many iterations will stop before the x values reaches 1 due to an intermediate value being seen somewhere. Many other iterations won’t even run because the original value will have been seen in the saved list.

3. There are no patterns in the way the x values converge to the final value of 1. This can be seen in the above image where the function was run for n = 100. For x = 97, there are a lot more intermediate values than there are for the larger x value of 99. Also, a smaller x value, 94, is seen to not have any intermediate values at all, since 94 appears in the iteration where x = 97. The intermediate values have a structure where for one iteration they will reach the maximum value for their subset of consecutive numbers. Every iteration after that will have significantly less intermediate values. Finally, not every terminating run will pass through 2. Only the first iteration will pass through 2 since the final iterations must follow the sequence ...8,4,2. Any other iteration will terminate once that sequence has started.