School of Computer Science, University of Windsor

60-141: Introduction to Algorithms and Programming II
Term: Summer 2014 (July-August)
Instructor: Dr. Asish Mukhopadhyay

Assignment 5
Posted: 1st August, 2014
Due: 8th August, 2014, 11:59pm

Preamble: This assignment has been designed to consolidate your understanding of dynamic data structures, queues in particular.

Grading Scheme: The problem is worth 10 points, 2 for program documentation, 2 for effort, 6 for correctness. Mail your solution to parti@uwindsor.ca before the due date. The submission should have the Assignment number, your name and student ID on it as well as on the subject line of the e-mail, with an attachment that contains a script file and a source file.

Credits: The conception of this lab is mine.

Problem: As a preparation for solving this problem, read carefully section 12.6 of your textbook (7th Edition) on implementing the queue data structure as a linked list, particularly the routines for maintaining a queue dynamically (see enqueue and dequeue routines of Fig. 12.13). You will use this data structure to implement `radix sort`, explained below.

Assume you are given n positive decimal integers, each consisting of, say 3 digits, such that the most significant digit is non-zero. Radix sort works like this. Maintain 11 queues, $q_0, q_1, q_2, \ldots, q_9, q_{10}$, the first 10 corresponding to each of the decimal digits from 0 to 9 and an 11th one, whose role is explained below.

Start with the least significant digit and go through the list of n numbers one by one. If this digit is d_0 ($0 \leq d_0 \leq 9$) for a considered number, then add (or enqueue) this number to the d_0-th queue. Once all the n numbers have been thus enqueued, enqueue all the numbers in the 11th queue q_{10} by dequeuing them from the queues $q_0, q_1, \ldots, q_9$, in this order (you have to check if a queue, q_i, is non-empty or not).

Now, repeat the above queuing process for the next significant digit, but this time considering the

numbers for queuing by this significant digit by dequeuing them one-by-one from the 11th queue. Repeat again, for the third and most significant digit. Now the 11th queue will have the numbers in sorted order, which you can output by dequeuing them one by one.

While I have explained the problem using numbers with 3 digits, your program should be able to handle the general case of numbers, each with d digits, $d \geq 1$, with the most significant digit greater than 0.

Consider the example case of sorting five 3-digit numbers, 121, 340, 132, 231, 312, by radix sort.

Step 1: The number 121 goes into the queue q_1, 340 into the queue q_0, 132 into the queue q_2, 231 into the queue, q_1, 312 into the queue q_2 (rest of the queues are empty).

Step 2. Dequeuing q_0, q_1 and q_2... in that order and enqueuing them in q_{11}, puts the numbers in this queue in the order 340, 121, 231, 132, 312.

Step 3: Dequeuing the numbers from q_{11} and adding these to an appropriate queue, according to the second significant digit, we have in q_4: 340, in q_2: 121, in q_3: 231, 132 and in q_1: 312 (rest of the queues are empty).

Step 4: Dequeuing from q_1, q_2, q_3 and q_4 in this order and enqueuing them in q_{11} we have in this queue the numbers in the order 312, 121, 231, 132 and 340.

Step 5: Repeating on the 3rd and most significant digit, q_1 has 121, 132, q_2 has 231 and q_3 has 312, 340. Dequeuing from q_1, q_2 and q_3 in this order has the numbers enqueued in q_{11} in the order: 121, 132, 231, 312, 340.

Step 6. Outputting from q_{11} now gives the sorted order: 121, 132, 231, 312, 340.