CS 2420-001 Algorithms and Data Structures

Spring Semester, 2014

Assignment 2: Linked Lists, Queues, and Stacks

Due Date: Friday, Feb. 7, 2013 (at the beginning of CS2420 class)

(**Note:** This assignment has only programming questions. Please submit your cpp files to Canvas.)

1. In this question we will implement the linked list we discussed in class. We want to use a linked list to maintain a sorted list of numbers in **descending** order (note that the example we used in class follows the ascending order). You need to write procedures to support the following operations. Each operation should take linear time in the worst case. **(30 points)**

   (a) $insert(x)$: insert a new number x to the list, and after the insertion, the new list should still be a sorted list (in descending order). For example, if the list is $123, 45, 32, 12, 6$, then after the operation $insert(40)$ the new list should be $123, 45, 40, 32, 12, 6$. But if x is already in the list, then you do nothing.

   (b) $remove(x)$: delete the number x from the list. If x is not in the list, then you do nothing. For example, if the list is $123, 45, 32, 12, 6$, then after the operation $remove(32)$ the new list should be $123, 45, 12, 6$.

   (c) $reverse()$: reverse the list. For example, if the list is $123, 45, 32, 12, 6$, then after the operation $reverse()$ the new list should be $6, 12, 32, 45, 123$.

   On Canvas, go to the following directory: homework/hw2/question1. There are a starter cpp file "hw2_Q1.cpp", a insert data file "hw2_Q1_insertData.txt", and a remove data file "hw2_Q1_removeData.txt". The program first reads the numbers in the insert data file and stores them in array A. Then, the program reads the numbers in the remove data file and stores them in array B. Next, we do insert operations on all numbers in A and then do remove operations on all numbers in B. Finally, we do a reverse operation and print the final linked list to an output file "hw2_Q1_output.txt".

   You task is to complete the three functions: $insert()$, $remove()$, and $reverse()$.

   Please follow the instructions we used in Assignment 1, e.g., submit a single cpp file to Canvas; change the file name by adding your A number at the end; do not change the input/output style in your submitted program. These instructions are also applicable to the next two questions.

2. In this question we will use a linked list to implement the "queue" data structure we discussed in class. The queue is used to maintain a sequence of numbers (not necessarily sorted) in a first-in-first-out manner. You need to write procedures to support the following operations. Each operation should take constant time. **(20 points)**

(a) $enqueue(x)$: add a new number to the **end** of the queue. For example, if the queue is $5, 79, 6, 34$, then after the operation $enququе(100)$ the new queue should be $5, 79, 6, 34, 100$.

(b) $dequeue(x)$: return the **first** number of the queue and remove it from the queue. For example, if the queue is $5, 79, 6, 34$, then after the operation $x = dequeue()$, the new queue should be $79, 6, 34$, and x will have the value 5.

On Canvas, go to the directory: homework/hw2/question2. There are a starter cpp file "hw2_Q2.cpp" and an input file "hw2_Q2_input.txt". The program first reads the numbers in the input file and stores them in array A. Then, we do enqueue operations on all numbers in A. Afterwards, we do dequeue operations for ten times and store the results in another array B. The numbers in B will finally be output into an output file "hw2_Q2_output.txt".

You task is to complete the two functions: $enqueue()$ and $dequeue()$.

3. In this question we will use an array to implement the "stack" data structure we discussed in class. The stack is used to maintain a sequence of numbers in a last-in-first-out manner. You need to write procedures to support the following operations. Each operation should take constant time. **(20 points)**

   (a) $push(x)$: add a umber x on the top of the stack. For example, if the stack is $5, 79, 6, 34$ and 5 is on top, then after the operation $push(100)$ the new stack should be $100, 5, 79, 6, 34$ and 100 is on top. Note that if the stack is already full before the push operation, do nothing.

   (b) $pop()$: return the number on top of the stack and remove it from the stack. For example, if the stack is $5, 79, 6, 34$, then after the operation $x = pop()$, the new stack becomes $79, 6, 34$, and x will have the value 5. Note that if the stack is already empty before the pop operation, do nothing.

   On Canvas, go to the directory: homework/hw2/question3. There are a starter cpp file "hw2_Q3.cpp" and an input file "hw2_Q3_input.txt". Each line of the input file is the style of either "push x" or "pop": the former means an operation $push(x)$ and the latter means an operation $pop()$. The program reads the input file line by line and perform the operations accordingly. For each pop operation, the returned number will be stored in an output file "hw2_Q3_output.txt".

   You task is to complete the two functions: $push()$ and $pop()$.

   Note that if you want to take a look at the input file in Windows system, please use "WordPad" instead of "Notepad" to open it.

Total Points: 70