CSE331 – Project 3: Linked List

In this project you will complete an implementation of the Linked List data structure and use it to solve a modified *Josephus problem*. The project must be submitted to Handin no later than 11:59 pm Wednesday February 6, 2013. Email the TA with any questions. Good Luck!

Project Description

You will be solving a modified version of the Josephus problem from your book (page 108, exercise 3.6). The Josephus problem is the following game: N people, numbered 1 to N, are sitting in a circle. Starting at person 1, we pass a hot potato M times. The person holding the hot potato after M passes is eliminated. The next person picks up the hot potato and passes it again, this time in opposite direction. The last person remaining at the end is the winner. M is always $0<M\leq N$ and M may be either constant or random.

The code to start the project, and an example Linux executable, may be found in ~/CSE331/Projects/Project3/. The executable takes two arguments at the command line: n and m. Were n is the number of players, and m is the number of steps. If m is a negative number, than it's absolute value will be used to seed a random number generator, which will generate a new value of m each turn.

You must complete the implementation a program to solve our modified Josephus problem using a linked list. You are provided with a header file for a templated doubly linked list. You must complete the implementation of all of the public methods in RevList.h. You may neither add nor modify the way any of the public methods in RevList.h interact with the rest of the program. You may find it useful to add some private methods to RevList.h for functional abstraction. Also, reverse, delete and other public methods in RevList.h should run in constant time (O(1)) except for copy constructor, clear, destructor, and find() (that includes the running time of calls to private methods). After you've done that, complete the appropriate sections of the driver to come up with the correct permutation for our modified Josephus problem. Compare your solution to the sample run file.

Programming Notes

All of the input, output, and problem set up has already been completed in main.cpp, all you need to do is complete the while loop, which pushes the number of each player onto the vector v_order as they are eliminated.

If you wish to you may use a head node in your linked list, though it is not required. If you choose to use a head node:The next() method should never return a pointer to the head node, it should simply skip it and return the node after the head. If the list is empty, and the head node is the only thing left, simply return null. Likewise delete() should never delete the head node. If delete does receive the head node to delete it should return without doing anything.

Once again, the list class is templated and should work with any class (int, string, double, or a user-defined class). I will be testing for this when I grade the project.

If you have trouble getting the correct solution, try writing your own driver for the linked list and make sure it is functioning correctly. Once you have established that it is working correctly, you can move on to debug the driver.

Project Deliverables

The following files must be submitted via Handin no later than 11:59 pm Wednesday February 6, 2013:

- RevList.h – contains your implementation of a templated doubly linked list
- main.cpp – your completed implementation of the Josephus problem
- project3.pdf – files containing your answers to the written questions. (Must be a Portable Document Format file)

Written Questions

Submit the answers to these questions in a PDF file, called project3.pdf, along with your source code files. The versions of MS Office and OpenOffice in the CSE labs both support exporting to PDF format.

1. Each of your responses should give bounds on the running time in terms N and M. Were N is the number of players, and M is the number of passes. Justify each of your answers in paragraph form.
    a. Assuming a constant M, what is the upper bound on the running time of our version of the Josephus problem?
    b. Assuming a random M, what is the upper bound on the running time of our version of the Josephus problem?
2. Write an algorithm, in pseudo code, to reverse a *singly* linked list in O(N) time. Justify your answer in paragraph form.
3. Give an upper bound on the running time for each of the following operations, justify your answer in paragraph form.
    a. Deleting a key from a singly linked list
    b. Deleting a key from a doubly linked list
4. Let A and B be sets (a collection of unique items) expressed using linked lists. Provide pseudo code for computing A UNION B in O(1) time, where the result is the union of the two sets. Assume the sets are disjoint (have no elements in common). You may destroy A and B in the process.
5. Repeat 4, but assume the sets are not disjoint. The union operation must then remove any duplicate items. What is the best running time you can accomplish (asymptotically) for this operation?