Computer Science II — Homework 5 — Linked Lists and the Josephus Problem

Overview

This assignment is due Thursday, February 28 at 11:59:59pm and is worth 70 points toward your homework grade.

The Josephus problem is a legendary means of picking a survivor from a group of n individuals. The idea is that the n individuals are formed into a circle and an integer $m \geq 1$ is chosen. Starting at a designated spot, the individuals count off from 1 to m. The mth individual is "removed" from the circle, and the process repeats, starting from the next individual, not from original spot. This repetition continues until one individual is left. For example, if Brian, Sue, Andrew, Keisha, Erica, Tom and Paula are arranged in circle, with $m == 3$ and Brian at the designated starting location, then the order of removal is Andrew, Tom, Sue, Paula, Erica and Brian, with Keisha remaining as the survivor.

Your job will be to implement two solutions to the Josephus problem, one using a circularly linked list (the last node points to the first) of your own design and implementation (i.e. not using `std::list`) and a second using a `std::vector`. You should start counting the first time from the first name that you read from the input file. In each case, the individuals must be actually removed from the list (or vector), so that the size of the structure shrinks, eventually to one. Both solutions will be implemented together in a single main program.

Input / Output

The names of the individuals to form the circle will be input from a file, with each individual being a single string. No names will be repeated in this file. A second file will contain a sequence of positive integers giving different values of m to try. (This implies that the same list or vector must be used more than once, and copies must therefore be made.) For each value, the program should output the results from both the list and vector implementations of the algorithm. In each case, the survivor should be first, on a separate line, followed by the names of the individuals in the order they were removed, with up to 8 names per line.

Additional Requirements

You will not need a class for the vector version of the solution, but you should write a class for the linked-list version, allowing you to encapsulate details. My implementation uses just a few public member functions, plus several more private ones. Your class for the linked-list version will make use of a list-node class, which may be very simple and include public member variables. Please note that the list-based version of the solution may only use strings in addition to the linked-list functionality that you implement — it may not use `std::list` or `std::vector`.

Circularly-Linked Lists

You will need to design and implement several functions to operate on circularly-linked lists and you will need to alter others discussed in Lecture 10. You should start by writing and testing functions that find the "end" of the list, and that `insert` (at the end) and `remove` nodes. You will also need to copy entire lists (so that you do not destroy the ordering of the original list and you can reuse it). In addition, you will need to be able to destroy a list. Finally, since debugging linked-lists is particularly difficult, you should write a function to print the contents of a list. You may then call this function at many points in your code to ensure things are working correctly.

Submission

Please place all of your source code files and readme.txt file in a folder named hw5 (no spaces, no uppercase, not hw5_submit, not HW5). Make sure none of your file names include spaces. Zip your folder prior to submission, and submit it through the course web page.