Algorithms

CMPT 435

Assignment 1 - 75 points

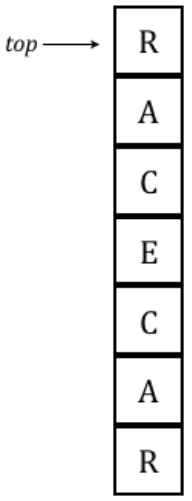
Goals

- to program a few elementary data structures and implement a few sorting algorithms

Requirements and Notes

- Develop and test a Node class. Use it to develop a stack, then to develop a queue. You must implement these yourself; you **may not** use any built-in features of the language or its libraries for the stack or queue. (You **may** use language libraries for other things, like file operations, etc.) [5 points]

- Download the the text file `magicitems.txt` from our web site.
- Read it line-by-line into **array**.



Stack

- Check each line of the array to see if it's a palindrome, ignoring spaces and capitalization. Print it out **only** if it is a palindrome. [5 points]
  - To check whether or not a given string is a palindrome, take it character by character (ignoring spaces and capitalization) and push each character on a stack and enqueue each character in a queue. When every character is on the stack and in the queue, pop the stack and dequeue the queue one character at a time. If they always match then the string is a palindrome.

- Develop **your own** implementation of selection sort, insertion sort, merge sort, and quick sort. Then sort `magicitems.txt` using each sort, printing the number of comparisons each time. Be sure to shuffle the array before each sort. To do that, write your own O(n) shuffle routine based on the Knuth shuffle (also known as the Fisher–Yates shuffle, but not known as the Rosanna shuffle, which is different). [15 points]

- Document all code (with line numbers and explanations) and your results (in a table) in a LaTeX document. Make multiple test runs for each sort and average the results. For the code, explain the good parts of how each works, as if you were teaching it to someone else, referencing line numbers for pedantic clarity. For the sort results, note the asymptotic running time of *each* sort and explain **why** it is that way. [50 points]

Your code must … [−∞ if not]

- **not** be in a package; that just makes it harder for me to compile and test
- separate structure from presentation, be professionally formatted yet uniquely yours (show some personality), and demonstrate best practices

Resources

- Linked lists are described in the 3rd edition of our text in chapter 10.2, starting on EC.
- Stacks and queues are described in the 3rd edition of our text in the beginning of chapter 10, starting on page 1110 1000.
- Insertion sort, merge sort, and quick sort are described in the 3rd edition of our text in sections 2.1, 2.3, and 7.1 respectively.

Submitting Your Work

Make **many** commits to GitHub. I do not want to see one massive "everything" commit when I review your code. (It's −∞ if you do that.) Commit early and often. And make sure your commit messages are descriptive and amusing.