CSAI – 230 - Fall 2024

Assignment 2: Stacks and Queues

Assigned: Sunday, October 27th in Class

Due: Thursday, November 7th at 17:00

Delayed submission with penalty until Saturday, November 9th at mid-night.

Goals

This assignment is an individual assignment and you will work on it on your own. The goal of this assignment is to be able to use stacks and queues, and to master and have an in-depth understanding of such primitive ADTs. In this assignment you will build a more complex ADT using the primitive stack and queue ADTs. Moreover, an important goal of this assignment is to be able to analyze an ADT that you will build yourself using more primitive ADTs, and to be able to compute the time and space complexity using the Big-O notation. Based on that you are asked also to enhance your first implementation to achieve better performance.

Details

In this assignment you are required to build a Queue ADT using a Stack ADT. You are required to use the linked list stack implementation we presented in class and included in the lecture slides. You are required to use exactly two stack instances in your Queue ADT implementation. **Note: the implementation in the slides might have some minor flaws, implementation overheads, and missing parts, as it was designed for presentation purposes, and you need to fix, modify, and enhance anything you see important to reach better results.**

In this assignment you are required to perform two attempts. In the first attempt, you are required to use the linked list stack implementation as is, and you are not allowed to change it, which will not result in a very optimum implementation; **basically you are not allowed to use except the push and pop methods**. Nevertheless, your implementation of the new Queue ADT should be as optimum and efficient as possible within these boundaries; using the linked list stack implementation as is.

In the second attempt, you will be allowed to introduce changes to the stack implementation to provide better performance and hence better running time; **but still you are not allowed to use except the push and pop methods** . It is mandatory to utilize all object oriented techniques, such as inheritance, to minimized the amount

of amendments as much as possible, and to provide copy and move constructors if needed to reduce memory operations overhead.

The new ADT Queue implementations in both attempts should account for any size and any data item type, thus you are required to utilize templates. Moreover, you are required to implement all data validations needed and handle/report errors whenever needed.

Finally, You are required to analyze the new Queue ADT, in both attempts, using the Big-O notation for the new Queue ADT operations, precisely the enqueue and dequeue. Moreover, you need to show that your second attempt results in a better running time than the first one.

What to submit

1. Your full in-line documented source code; it should include a main program that has some test cases that shows how your ADT works.
2. A PDF report that includes:
    a. Any assumptions you have made.
    b. Your design approach to solve this problem and why do you think it is the most optimum approach.
    c. Your Big-O Analysis with enough reasoning on how you reached such analysis.
3. A make file for building all your sources.
4. A readme file that explains how to compile and run your program.
5. Your code should compile using the GNU C/C++ compiler and run on Linux; I advice you to use a Docker image for that. Docker is a very important technology that utilizes process level virtualization and the GIT repository management system and can integrate with github. The benefit of using dockers to submit your assignment is mainly the ability of the TA to reproduce your work easily, and put you in the safe side of any assumptions about your work. (Let me know if you need help and if you are interested in learning how to use dockers I can help you with that during my office hours)

How to submit:

Compress all your work: source code, report, readme file, and any extra information into a zip archive. You should name your archive in the specific format <Student_ID>_<Name>_Assignment2.zip. Finally, upload your code to moodle.

If you were able to package your work into a docker environment, which is optional, you can mention your docker repository in your

readme file, but it is a must to upload all your source code assignment material to moodle.

Grade

This assignment is worth 5% of the overall course grade. The assignment will be graded on a 100% grade scale, and then will be scaled down to the 5% its worth. The grading of the assignment will be broken down as follows:

1. 10 % for just submitting a meaningful assignment before or on the due date. This 10% does not account for the correctness of your assignment but submitting an empty or incomplete assignment without code will definitely results in loosing this 10% and consequently the whole grade of this assignment.
2. 50 % for the correctness and the quality of your code.
3. 15 % for the quality of your inline documentation and the readme file.
4. 25 % for the report and the analysis.

Delays

You have up to 2 working days of delay, after which the assignment will not be accepted and your grade in that case will be ZERO. For every day (of the 2 allowed days), a penalty of 10% will be deducted from the grade. And of course you will lose the 10% mentioned in point 1 above under the “Grade” section.