The Programming Assignment Report Instructions
CSCE 221

1. The description of an assignment problem.

   The purpose of this assignment was to create a C++ program that would allow a user to enter an expression with or without variables in infix form. The simple calculator would then evaluate the expression by utilizing a stack, a que, and postfix form. The stack and que also used the templated linked list class created in part 1.

2. The description of data structures and algorithms used to solve the problem.

   (a) Provide definitions of data structures by using Abstract Data Types (ADTs)

   - Templated Doubly Linked List
   - Templated Linked Que
   - Templated Linked Stack
   - String
   - Vector
   - Token

   (b) Write about the ADTs implementation in C++.

   Notice that during the evaluation process of an expression the stack data structure should be used.

   - Templated Doubly Linked List
   - Templated Linked Que
   - Templated Linked Stack
   - String
   - Vector
   - Token

   (c) Describe algorithms used to solve the problem.

   - $Parser :: toPostfix()$
   - $Token :: get_operator_weight()$
   - $Evaluator :: evaluate$

- $Evaluator :: getValue()$

(d) Analyze the algorithms according to assignment requirements.

- $Parser :: toPostfix()$
- $Token :: get_{o}perator_{w}eight()$
- $Evaluator :: evaluate$
- $Evaluator :: getValue()$

3. A C++ organization and implementation of the problem solution

(a) Provide a list and description of classes or interfaces used by a program such as classes used to implement the data structures or exceptions.

- Parser
- Evaluator
- LinkedQue
- LinkedStack
- RuntimeException

(b) Include in the report the class declarations from a header file (.h) and their implementation from a source file (.cpp).

(c) Provide features of the C++ programming paradigms like Inheritance or Polymorphism in case of object oriented programming, or Templates in the case of generic programming used in your implementation.

Templated

- LinkedQue
- LinkedStack

Inheritance

- RuntimeException

4. A user guide description how to navigate your program with the instructions how to:

(a) compile the program: specify the directory and file names, etc.

(b) run the program: specify the name of an executable file.

5. Specifications and description of input and output formats and files

   (a) The type of files: keyboard, text files, etc (if applicable).

       No input or output files

   (b) A file input format: when a program requires a sequence of input items, specify the number of items per line or a line termination. Provide a sample of a required input format.

   (c) Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)

6. Provide types of exceptions and their purpose in your program.

   (a) logical exceptions (such as deletion of an item from an empty container, etc.).

       - QueueEmptyException
       - StackEmptyException

   (b) runtime exception (such as division by 0, etc.)

       - *DivisionByZeroException*() in evaluator.h

7. Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output.