CSCE 3110 Data Structures
Assignment 2

Introduction

- For short answer and multiple choice section(1-2), you need to provide clear, correct, and complete explanations.
- For the codding section(3), please write a brief file with code with screen-shot of execution result where you explain all the information required for the grader to grade your program, and complete comment on how to compile and execute your program.
- Ensure that your program compiles and executes at any CELL machine. Please note, if your program does not compile at CELL machine, your will NOT get any credits.
- On the top of your Word file or your Hand-written document, write a comment statement including your name & UNT ID, the course number, and Assignment on the top.
- Name your file as $Asgn2\_LastName\_FirstName.docx$ or $Asgn2\_LastName\_FirstName.pdf$.
- Upload your file to the Assignment 2 on Canvas.

1 Question 1 (20 points)

Find two functions $f(N)$ and $g(N)$ such that neither $f(N) = O(g(N))$ nor $g(N) = O(f(N))$.

2 Question 2 (20 points)

Give an analysis of running time (Big-Oh)

a.

```
sum = 0;
for( i = 0; i < n; ++i )
    for( j = 0; j < i; ++j )
        ++sum;
```

b.

```
sum = 0;
for( i = 0; i < n; ++i )
    for( j = 0; j < i; ++j )
        for( k = 0; k < j; ++k )
            ++sum;
```

3 Question 3 (60 points)

Implement a stack and solutions to the following problems:

1. Balancing parentheses
2. Evaluating postfix expressions
3. Transforming infix expressions into postfix expressions

3.1 Programs

We are providing some sample code and input files:

- `public/`
- `stack.cpp`
  — stack implementation
- `stack.hpp`
  — stack header file
- `balancing.cpp`
  — main method to check whether an expression is balanced
- `input_balanced.txt`
  — test cases for `balancing.cpp`
- `infix2postfix.cpp`
  — main method to transform an infix expression into postfix
- `input_infix2postfix.txt`
  — test cases for `infix2postfix.cpp`
- `postfixEval.cpp`
  — main method to evaluate postfix expressions
- `input_postfixEval.txt`
  — test cases for `postfixEval.cpp`

3.2 Implementation

To compile, run:

```
$ g++ stack.cpp balancing.cpp
$ g++ stack.cpp infixtopostfix.cpp
$ g++ stack.cpp postfixEval.cpp
```

To run each program, run:

```
$ ./a.out
```

3.3 Notes

- `balancing.cpp` must balance parentheses, square brackets, and curly braces: (), [], {}.
- `infix2postfix.cpp` converts infix (midfix) expressions into postfix expressions using stacks.
- `postfixEval.cpp` computes the value of a postfix (suffix) expression, supporting arithmetic operations on postfix expressions.
- The test cases follow this format: *expected_solution input*. Given the input, your task is to implement code that derives the *expected_solution*, rather than simply printing it.
- While we provide a few test cases, you are expected to create additional ones to ensure your code functions correctly.

3.4 Grading

- Implementing the stack: 10 points
- Solving the balancing problem: 10 points
- Solving the evaluating postfix expressions: 20 points
- Solving the transforming infix expressions into postfix: 20 points