CSCE 221 Cover Page
Homework Assignment

First Name: Jialu Last Name: Zhao UIN: 426000712

User Name: zhaojialu123 E-mail address: zhaojialu123@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources	Slides		
People	Dr.Leyk		
Web pages (provide URL)	http://courses.cs.tamu.edu/teresa/csce221/pdf-lectures/parser.pdf		
Printed material			
Other Sources			

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Jialu Zhao Date 16/03/2017

Programming Assignment 3 Report (Part 2,3)

1.The description of an assignment problem.

Part 2:

implement templated stack and queue data structures using the templated linked list class.

Part 3:

(a)implement the class parser with a function to_postfix() to translate input infix form of an algebraic expression into its postfix form using stack and queue as the auxiliary data structures.

(b)implement the class evaluator with a function evaluate() to get the value of an algebraic expression based on its postfix form.

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

(design of part 2 and part 3)

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

Abstract Data Type(ADT) that specifies the type of the data stored operations that support the data. In part 2, we use templated doubly linked list to help implement templated stack and queue data structures. In part 3, we use templated linkedstack and linkedqueue which built in part 2 to help implement class parser which can translate input infix form of an algebraic expression into its postfix form. And we also use another templated linkedstack to implement the class evaluator which get the value of an algebraic expression based on its postfix form.

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

(Describe each class private and public members)

Part2:

(a) LinekdQueue:I use Templated DoublyLinkedlist's constuctor and desctructor to implement LinekdQueue's constructor and destructor. There are also copy constructor and copy assignment which will be used to help implement class parser and class evaluator. isEmpty function, size and first function directly call function isEmpty, size and first function of doublylinkedlist. Also, I use removefirst function of doublylinkedlist to implement dequeue function in linkedQueue and use insertlast function of doublylinkedlist to implement enqueue function in linkedQueue. For the << operator, I use copy constructor and dequeue function to output every elements one by one.

(b) LinekdStack:I use Templated DoublyLinkedlist's constuctor and desctructor to implement LinekdStack's constructor and destructor. There are also copy constructor and copy assignment which will be used to help implement class parser and class evaluator. isEmpty function, size and top function directly call function isEmpty, size and first function of doublylinkedlist. Also, I use removefirst function of doublylinkedlist to implement pop function in linkedStack and use insertlast function of doublylinkedlist to implement push function in linkedStack. For the << operator, I use copy constructor and pop function to output every elements one by one. There is a little difference from LinkedQueue, because stack is the structure that first in and last out, I use two helper Stack to get the right sequence of the elements and output them.

Part3:

(a)Class Parser: I use a LinkedStack of string: opStack and a linkedQueue of string: expQueue as private member to build my topostfix function. I also have vector infix and string IF as private memeber to build getToken function. Private memebr function: isoperand, isoperator, inputpriority and stackpriority, hashigherPrecedence, ParanthesesBalanced, InvaildInput function. public function: topostfix, printinfix and printpostfix function.

(b)Class Parser: I use a LinkedStack of double: valStack and two linkedQueues of string: helper and postfix as private member to build my getvalue function. Helper was built to help convert value from parser to evaluator. Public function: usersvalue was built to prompt users to input the value of variables.

(c)Describe algorithms used to solve the problem.

(Algorithms and their implementations)

```
(1)
Algorithm Parser:
input: a infix string
output: a postfix string
string infix (no invaild input and if
Parantheses are Balanced or not)
string infix -> vector<string> infix -> LinkedQueue infix
LinkedQueue infix -> LinkedQueue postfix
(2)
Algorithm Evaluator:
input: a postfix LinkedQueue
output: a double result
LinkedQueue postfix -> get value from user -> double result
```

(d)Analyze the algorithms according to assignment requirements.
(1)Class Parser:

```
bool IsOperand(string); --O(1)
bool IsOperator(string); --O(1)
int  inputPriority(string); --O(1)
int  stackPriority(string);  --O(1)
int HasHigherPrecedence(int op1Weight, int op2Weight); --O(1)
bool ParanthesesBalanced(); --O(n)
bool InvaildInput(); --O(n)
void getToken(); --O(n)
Parser(string s); --O(1)
~Parser(); --O(1)
LinkedQueue<string> ToPostfix(); --O(n)
```

(2)Class Evaluator:

```
Evaluator(Parser& par); --O(1)
void usersValue(); --O(n)
double getValue(); --O(n)
```

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.

```
LinkedStack class   <string> and <double>
LinkedQueue class   <string>
RuntimeException class
string class
vector class
math class
```

(b)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.

Inheritance: I use LinkedStack and LinkedQueue to implement my class Parser and Evaluator.

Classes: Use several classes to create user defined data types.

Data Abstraction and Encapsulation: Use templated DoublyLinkedList to implement LinkedStack and LinkedQueue.

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.

compile: type "make " in the terminal or putty

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

run : type "./main" in the terminal or putty

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

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

(1)After seeing the instructions "please input the value of a:", type the specific value of the variable.

(2)After seeing the instructions " Please input an infix expression from keyboard:", type any formula you want and get its value. note: Please use # as an end notation everytime.

For example:

please input the value of a:

4

please input the value of b:

2

The associative value for this expression is:

7

Please input an infix expression from keyboard:

a+6#

Infix expression is: {a+6}

Convert to its postfix expression is: {a6+}

please input the value of a:

3

The associative value for this expression is:

9

(b)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.)

when user type an invaild formula like ")jk", an exception: Stack is empty will be thrown which is not expected.

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.).

(1) When the parantheses in the formula are not balanced, an excetion will be thrown, inform user that the paranethese are not balanced.

(2) When there is an invaild input like ! or ~, an exception will be thrown, inform user that there is an invaild input in the formula.

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

When LinkedStack or LinkedQueue are empty, a runtime exception will be thrown.

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.

Test for LinkedQueue(string):

```
Test for LinkedQueue
queue = {1223334444555555123456}
first of queue = 1

dequeue every elements one by one from the queue:
1
22
333
4444
55555
123456
After dequeueing all elements from queue, queue = {}
```

Test for LinkedStack(string):

```
Test for LinkedStack
stack = {ORRAAANNNNGGGABCDE}
top of stack = ABCDE

pop every elements one by one from the stack:
ABCDE
GGG
NNNN
AAA
RR
O
After popping all elements from stack, stack = {}
```

Test for LinkedQueue(double):

```
Test for LinkedQueue
queue = {0.811.21.9}
first of queue = 0.8

dequeue every elements one by one from the queue:
0.8
1
1.2
1.9
After dequeueing all elements from queue, queue = {}
```

Test for LinkedStack(double):

```
Test for LinkedStack
stack = {0.811.21.9}
top of stack = 1.9

pop every elements one by one from the stack:
1.9
1.2
1
0.8
After popping all elements from stack, stack = {}
```

Test for Parser and Evaluator:

```
Test for Parser
{a+(6/b)}
{a6b/+}
Test for Evaluator
please input the value of a:
4
please input the value of b:
2
 The associative value for this expression is:
7
```

Test for user menu:

```
Please input an infix expression from keyboard:
(a+8)#
Infix expression is:
{(a+8)}
 Convert to its postfix expression is:
{a8+}
please input the value of a:
3
 The associative value for this expression is:
11
```

Test for invaild input:

```
Please input an infix expression from keyboard:
a!0
Infix expression is:
 Invaild input!!!
```

Test for unbalanced parenthese:

```
Please input an infix expression from keyboard:
(3+8#
Infix expression is:
Unbalanced parantheses!!!
```