**C++ Workshop – 150018**

**Homework Assignment #8**

**Stacks and Queues**

1. In this exercise, we want to write a program that will input a mathematical expression (infix) as a string from the user, and calculate the value of the expression while maintaining the order of mathematical operations. This is done using a stack. The implementation of class stack for int can be copied from the course handbook.  
   1. Write a global function named **infixToPostfix** that gets an expression in infix representation and returns a new string representing the expression but this time in the postfix form.  
       (The algorithm is given below). For example, for the input **"(5+3)\*((20/10)+(8-6))"** the function returns **"5 3 + 20 10 / 8 6 - + \*"**

Note:

* + - The quotation marks are not part of the expression
    - Expressions can contain numbers that have more than one digit.
    - Add a single space between any two components in the postfix expression that is returned

**The algorithm to convert infix to postfix**

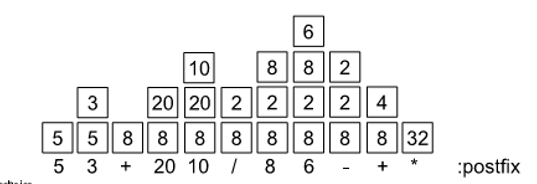
This following algorithm accepts a string in infix and one character after another transforms it into a postfix form. The expression that we receive as input may have round parentheses only.

1. Create an empty stack 🡪 str
2. Create an empty stack of chars 🡪 stk
3. Read the first character from input 🡪 ch
4. As long as there still is input,do:
   1. if ch is an opening(left) parenthesis then push it on the stack
   2. if ch is a closing(right) parenthesis, then pop the contents of the stack until (not including) the opening(left) parenthesis and append to str. Then pop the opening(left) parenthesis.
   3. if ch is an operator pop from the stack all operators with higher precedence and append them to str. At the end push ch onto the stack.
   4. if ch is a digit, append it to str
   5. read the next character from the input 🡪 ch
5. As long as the stack is not empty, pop the stack and append to str
6. Write a global function called **CalcPostfix** that receives as a parameter a string representing an expression in postfix format. The expression might include add, subtract, multiply, and divide operations . The function must calculate and return the result of the expression. (The algorithm given soon).

For example, for input **5 3 + 20 10 / 8 6 - + \*** thefunction will return 32.

**The algorithm for computing a postfix arithmetic expression using a stack:**

1. start with empty stack
2. iterate over the elements of the expression from left to right:
   1. if next element is an operand add it to the stack
   2. if it is an operator execute the operator on the two top elements of the stack and push the result onto the stack as seen in the figure below.



1. **attention**: in the function infixToPostfix the stack will used for holding temporary char values, and in the function calcPostfix it will be used for float type. There is no need to copy the implementation for char separately since the compiler can use the int implementation from char to int and back according to the ascii value.
2. Use the following main program to check the correctness of the program you have written.

int main()

{

string exp;

cout << "enter an infix expression as a string" << endl;

cin >> exp;

string postfix = infixToPostfix(exp);

cout << postfix<<endl;

cout << calcPostfix(postfix)<<endl;

return 0;

}

1. In class we defined the abstract class Queue:

class Queue

{

public:

virtual void clear() = 0;

virtual void enqueue(int value) = 0;

virtual int dequeue() = 0;

virtual int front () = 0;

virtual bool isEmpty() const = 0;

};

We gave 2 different implementations for a queue, one using an array and the other using a list. In this exercise we would like to give the queue a third implementation, this time using a stack (and another helper stack).

class QueueStack : public Queue

{

protected:

stack\* data;

public:

QueueStack();

void clear() override;

int dequeue() override ;

void enqueue(int value) override;

int front() override;

bool isEmpty() const override;

};

You need to implement this former class.

Use the following main program to check the correctness of the class you wrote.

#include <iostream>

#include " QueueStack.h"

using namespace std;

int main() {

Queue\* Q;

Q = new QueueStack();

try {

for (int i = 0; i < 10; i++)

Q->enqueue(i);

}

catch (const char\* msg)

{

cout << msg;

}

cout << "first on Q is: " << Q->front() << endl;

cout << "take out 2 elemets:" << endl;

cout << Q->dequeue() << ' ' << Q->dequeue() << endl;

cout << "first on Q is: " << Q->front() << endl;

Q->enqueue(8);

Q->enqueue(9);

while (!Q->isEmpty())

cout << Q->dequeue() << " ";

return 0;

}

GOOD LUCK!