744EJERCICIOS CON PILAS Y FILAS

OBJETIVOS

Durante esta actividad, los alumnos serán capaces de:

- Resolver e implementar diversos problemas de programación en C++ utilizando pilas y filas.

Esta actividad promueve las siguientes habilidades, valores y actitudes: análisis y síntesis, capacidad de resolver problemas, creatividad, y uso eficiente de la informática y las telecomunicaciones.

DESCRIPCIÓN DE LA ACTIVIDAD

Esta actividad puede ser elaborada de manera individual.

Escribe una clase llamada **`StacksAndQueues`**. En esta clase deberás colocar únicamente los métodos que resuelvan los problemas que se describen a continuación.

En la parte superior del archivo coloca en comentarios los datos personales de los autores de la tarea. Por ejemplo:

```
/*----------------------------------------------------------------
 * Actividad de programación: Listas encadenadas
 * Fecha: 14-Oct-2015
 * Autor:
 *        1160611 Anthony Stark
 *----------------------------------------------------------------*/
```

Para el problema 3, puedes auxiliarte del método **`tokenize`** que recibe una cadena y la separa en sus elementos básicos, colocándolos en una fila de cadenas. Por ejemplo, si la entrada es la cadena **`"123 34 7+*-"`**, entonces devuelve una fila con los siguientes elementos: **`"123"`**, **`"34"`**, **`"7"`**, **`"+"`**, **`"*"`** y **`"-"`**.

```
#include <iostream>
#include <string>
#include <cctype>
#include <queue>

using namespace std;

queue<string> tokenize(string str) {
     int i = 0;
     int length = str.size();
     string aux;
     queue<string> result;

     while (i < length) {
             if (isdigit(str[i])) {
                     aux.clear();
                     do {
```

```
                        aux += str[i];
                        i++;
                } while(isdigit(str[i]));
                result.push(aux);
        } else if (isspace(str[i])) {
                i++;
        } else {
                aux.clear();
                aux += str[i];
                result.push(aux);
                i++;
        }
    }
    return result;
}
```

Estos son los métodos que debes implementar:

1. **`bool balancedBrackets(const string &expr)`**

   Devuelve **`true`** si la cadena **`expr`** que recibe como parámetro contiene una expresión en donde todos sus símbolos de agrupación (paréntesis **`()`**, corchetes **`[]`** y llaves **`{}`**) están correctamente anidados y balanceados. Devuelve **`false`** en caso contrario. Se debe ignorar cualquier carácter de **`expr`** que no sea paréntesis, corchetes o llaves.

   Descripción del algoritmo: Empieza con una pila vacía. Recorre cada uno de los caracteres de **`expr`** de izquierda a derecha:

   - Si encuentras un símbolo de apertura (**`(`**, **`[`** o **`{`**) debes insertarlo en la pila.
   - Si encuentras un carácter de cierre (**`)`**, **`]`** o **`}`**) debes remover el carácter del tope de la pila y verificar que ambos caracteres hagan pareja. Debes terminar el algoritmo con **`false`** si los respectivos caracteres no hacen pareja, o si la pila estaba vacía antes de intentar remover el elemento del tope.

   Si al final la pila está vacía debes devolver **`true`**, de otra forma debes devolver **`false`**.

2. **`queue<int> merge(const queue <int> &q1, const queue<int> &q2)`**

   Devuelve una nueva fila con el resultado de efectuar el algoritmo de mezcla a partir del contenido de las filas **`q1`** y **`q2`**. Las filas **`q1`** y **`q2`** deben llegar ya ordenadas de forma ascendente.

   Descripción del algoritmo: Empieza con una fila resultante vacía. En cada iteración determina quien tiene al inicio el elemento x más pequeño de entre **`q1`** y **`q2`** (recuerda que tanto **`q1`** y **`q2`** están ordenadas de forma ascendente, por lo que los elementos más pequeños siempre estarán justo al inicio de cada una de estas filas). Remueve a x de la fila que lo contiene y añádelo al final de la fila resultante.

Las iteraciones terminan cuando alguna de `q1` o `q2` queda vacía. Posteriormente hay que copiar a la fila resultante todos los elementos que quedaron en la fila `q1` o `q2` que aún no está vacía. Finalmente debes regresar la fila resultante.

3. `string convertInfixToPostfix(const string &expr)`

Recibe como parámetro la cadena `expr` que contiene una expresión aritmética en notación infija. Devuelve una cadena con la expresión equivalente en notación posfija. No realiza ningún tipo de validación respecto a la expresión de entrada.

Descripción del algoritmo: Empieza con una pila vacía y una fila resultante vacía. Inserta todos los elementos de `expr` en otra fila (usando el método tokenize) y procesa dichos elementos en orden FIFO:

- Si encuentras un número, insértalo en la fila resultante.
- Si encuentras un paréntesis izquierdo `(`, insértalo en la pila.
- Si encuentras un operador ⊗ (donde ⊗ puede ser `+`, `-`, `*` o `/`), realiza lo siguiente:
    - Mientras que la pila no esté vacía y además el tope de la pila sea diferente al paréntesis izquierdo `(`, realiza lo siguiente:
        - Si el elemento del tope de la pila tiene mayor precedencia que ⊗, entonces hay que sacar dicho elemento de la pila e insertarlo en la fila resultante. De lo contrario se debe terminar el ciclo «mientras» más anidado.
    - Inserta ⊗ en la pila.
- Si encuentras un paréntesis derecho `)`, realiza lo siguiente:
    - Mientras que la pila no esté vacía y además el tope de la pila sea diferente al paréntesis izquierdo `(`, realiza lo siguiente:
        - Saca el elemento del tope de la pila e insértalo en la fila resultante.
    - Si la pila no está vacía, remueve el paréntesis izquierdo `(` del tope de la pila.

Si al llegar a este punto la pila no está vacía, remueve uno por uno todos elementos de la pila e insértalos en la fila resultante.

Finalmente, devuelve una cadena conformada por la concatenación de todos los elementos de la fila resultante, usando un espacio en blanco como separador entre elementos.

Consejo: Para el algoritmo anterior, puedes usar el siguiente método para determinar si el elemento del tope de la pila tiene mayor precedencia que el operador ⊗:

```
bool hasHigherPrecedence(const string &stackTop, const string
&operator) {
     return !((stackTop == string("+") || stackTop == string("-")) &&
              (operator == string("*") || operator == string("/")));
}
```

¿QUÉ SE DEBE ENTREGAR?

Sube el archivo **`stacksandqueues.h`** a Blackboard, en la sección de "Envío de tareas".

EVALUACIÓN

Esta actividad será evaluada utilizando los siguientes criterios:

100	La actividad cumple con todos los requerimientos.
-10	No se incluyó en comentario los datos del autor.
10	El programa fuente produce uno o más errores al momento de compilarlo.
50-90	El programa funciona, pero produce algunos errores a tiempo de ejecución y/o los resultados no son del todo correctos.
DA	La solución es un plagio.