Práctico 7

Implementación de Pilas y Filas

NOTA: Los ejercicios deberán entregarse completos, siguiendo los criterios aconsejados por la cátedra y con los controles adecuados.

Ejercicio 1 

Implementar para la librería dada en la cátedra (PilaA) la operación básica que permita intercambiar el contenido de dos pilas respetando el siguiente prototipo.

```
void swap(PilaA<T, LONG_MAX_PILA> &x)
```

Ejercicio 2  (Obligatorio)

Implemente un TDA que represente una pila usando solo punteros y respetando la declaración que se detalla a continuación.

```
template<class T>
class stack {
public:
   stack();
   ~stack();
   bool empty();     //retorna true si la pila está vacía y falso en otro caso
   unsigned int size(); //retorna la cantidad de elementos de la pila
   T top();       //retorna el elemento que se encuentra en el tope de la pila
   void push(T);  //coloca el elemento T en el tope de la pila
   void pop();
   // TODO: completar con operadores y métodos que considere necesarios
private:
   // TODO: completar
};
```

Nota: no usar arreglos, tipos de datos definidos en STL, ni las implementaciones de lista vista en la cátedra. El archivo tiene que llamarse **stack.h**.

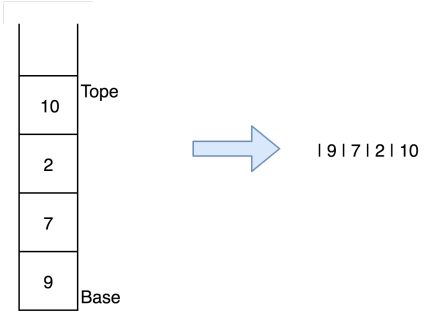
Documente brevemente dentro del código las decisiones que va tomando a medida que implementa el TDA Pila. Fundamente teniendo en cuenta el orden de los algoritmos, la utilización de memoria, facilidad de codificación y testing.

RECOMENDACIÓN PARA LOS EJERCICIOS 3 Y 4: se sugiere realizar primero la implementación con el `stack` provisto en STL. Una vez resuelto, probar que funciona con la librería `stack` implementada en el ejercicio 2.

Ejercicio 3 (Obligatorio)

Implemente una función de aplicación que imprima el contenido de una pila en el orden en que los elementos fueron insertados (el elemento de la base en el extremo izquierdo y el tope en el extremo derecho). Respetando el siguiente prototipo:

```
template <class T> void print(stack<T> s);
```



La función tiene que funcionar con la implementación del ejercicio 2 y la de STL.

1) ¿Por qué es importante implementar el constructor de copia en `stack` del ejercicio 2?
2) ¿Qué otros operadores considera apropiado implementar? En caso afirmativo programe los operadores correspondientes.

Ejercicio 4

Implemente una función de aplicación que convierta una expresión matemática en notación prefija a notación infija. Respetando el siguiente prototipo:

```
string preToInfix(string pre_exp)
```

En la expresión prefija se usa el símbolo espacio para separar números y operadores, por ejemplo

```
"+ 12 3"="(12+3)"
"+ 120 321"="(120+321)"
"+ - 120 3 21"="((120-3)+21)"
"* - A B / + C D * X Y"="((A-B)*((C+D)/(X*Y)))"
```

Nota: Asumimos que `pre_exp` es siempre una expresión sintácticamente correcta.

Ejercicio 5

En la librería de `FilaA` vista en la cátedra implemente los operadores `<<` (flujo de salida para imprimir la fila por pantalla) y los operadores de comparación `==` y `!=`.

Ejercicio 6  (Obligatorio)

En la librería `FilaA` vista en la cátedra implemente las siguientes operaciones básicas:

```
T recuperarUltimo();  //retorna el elemento que está al final de la fila
void quitarN(unsigned int n);  //quita los n primeros elementos de la fila, en caso de
                               //que n sea mayor a la cantidad de elementos la fila queda
                               //vacía.
void ponerAlFrente(T x); //pone el elemento x al frente de la fila
void invertir();         //invierte la fila
```

Es un requerimiento no funcional **optimizar la complejidad y el consumo de memoria** en los cuatro métodos.

Ejercicio 7 

Utilizando el TDA `FilaA`, implemente una función que genere los primeros N números binarios a partir del número 1 y devuelva los números en una lista de strings. Respetar el siguiente prototipo:

```
list<string> generarNBinarios(unsigned int n);
```

Ejemplos:

- si n = 5, entonces la salida es ["1", "10", "11", "100", "101"]
- si n = 7, entonces la salida es ["1", "10", "11", "100", "101", "110", "111"]

En el caso de que el parámetro n sea 0 retornar una lista vacía. **No usar algoritmos para convertir números decimales en binarios.**