ALGORITMOS Y ESTRUCTURAS DE DATOS

Trabajo Práctico no. 5 **Fecha: 21/04/22**

Tema: **Tipo de datos PILA**

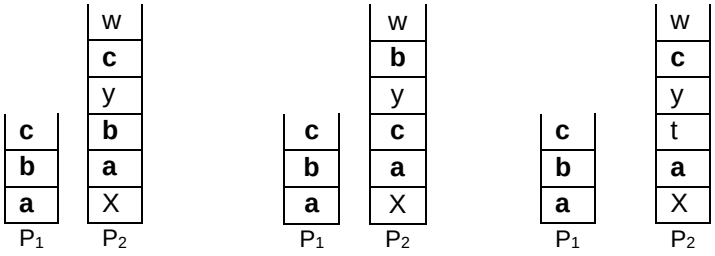
1. Considere el ADT PILA(item) visto en clase:

   a. Agregue a la ***especificación algebraica*** del mismo las siguientes operaciones:

      i. Una selectora: FONDO, que devuelve el item que se encuentra en el fondo de la pila.

      ii. Un test: INCLUIDA, que dadas dos pilas determina si los elementos de la primera pila se encuentran contenidos en la segunda pila, respetando el orden relativo de los mismos.

      Ejemplos:



   b. Como ***usuario*** del ADT diseñe una función recursiva **esCreciente** que dada una pila de números enteros determine si está ordenada de forma creciente desde el fondo hasta el tope.

   c. Codifique en un archivo de nombre **Pila.h** una implementación en lenguaje C del ADT PILA de enteros con lista enlazada utilizando la tipificación vista en la clase práctica. Escriba un programa para probar todas las operaciones de la Pila. Calcule la complejidad de las operaciones en notación O Grande

2. Teniendo en cuenta las operaciones del ADT PILA(item): PV (Pilavacia), Push, PushF, Pop, Top y Fondo indique en función de las constructoras primitivas cuál es la pila resultante en cada caso:

   a. **Push**(**Push** (**PV**,c), **Fondo**(**Push** (**Pop**(**PushF**(**Push** (**PV**,e), f)),g)))

   b. **Push**(**Push**(**Pop**(**Push**(**PushF**(**Push**(**PV**, a), b), c)),a) , **Top**(**Push**F(**Push** (**PV**, J), k)))

3. **Como usuario del ADT PILA** escriba el algoritmo para **CONVERTIR** una expresión aritmética dada en notación infija a una expresión en notación posfija*****. El proceso de convertir acepta una expresión infija como entrada y produce una expresión posfija como salida. Considere expresiones bien formadas que tengan variables (de la 'a' a la 'z'), operadores binarios (+, -, *, /), el operador unario (~) y paréntesis terminadas por la marca final '='. La idea es utilizar una pila para almacenar los operadores a medida que son encontrados para más tarde desapilar estos operadores de acuerdo a su precedencia.

Considere los siguientes ***casos de prueba***:

forma infija	forma posfija
a+b	a b +
~a+b	a ~ b+
a+b*a	a b a * +
(a+ (~b))*c	a b ~ + c *
a*(b+c)+a/e	a b c + * a e / +
a+b-c	a b + c -

forma infija	forma posfija
(a+(~b-c))	a b ~ c - +
(a+b)-c	a b + c -
(a-(b+c))	a b c + -
(b-a)/(c+d)	b a - c d + /
a+b/(d-a)*e	a b d a - / e * +
a*b/c	ab*c/