Programación Orientada a Objetos
Ejercicios de Colecciones Genéricas

1. Escriba una aplicación que pida al usuario una cadena de texto y utilice una pila para invertir el texto. Imprima el resultado.

2. Escriba una aplicación que utilice una pila para determinar si una cadena introducida por el usuario es un palíndromo. Debe ignorar la capitalización del texto.

3. Escriba una aplicación que tenga una función llamada **Unir** que reciba dos listas y las una en una nueva lista, la cual debe retornar. Utilice la función desde el Main e imprima su resultado.

4. Escriba una aplicación que tenga una función llamada **Depurar** que reciba una lista y elimine sus elementos repetidos. La lista debe pasarse por referencia.

5. Escriba una aplicación que tenga una función llamada **Intercalar** que reciba dos listas e intercale sus elementos en una nueva lista, la cual debe retornar. Utilice la función desde el Main e imprima su resultado.

6. Las pilas son utilizadas por los compiladores para evaluar expresiones y generar código máquina. En este ejercicio, investigaremos cómo los compiladores evalúan expresiones aritméticas que consisten únicamente de constantes, operadores y paréntesis.

   Los humanos generalmente escribimos expresiones como 3 + 4 y 7 / 9, en las cuales el operador (en este caso + o /) es escrito en medio de los operandos. A esto se le llama *notación infija*. Las computadoras "prefieren" la *notación posfija*, en la cual el operador es escrito a la derecha de sus dos operandos. Las expresiones infijas anteriores se escriben en notación posfija como 3 4 + y 7 9 / respectivamente.

   Para evaluar una expresión infija compleja, un compilador primero debe convertir la expresión a notación posfija y luego evaluar la versión posfija de la expresión. Cada uno de esos algoritmos requieren una sola pasada de la expresión de izquierda a derecha. Cada algoritmo utiliza una pila como auxiliar para realizar sus operaciones, y en cada algoritmo la pila se utiliza para un propósito diferente. En esta parte, usted implementará el algoritmo de conversión de infijo a posfijo. En el siguiente ejercicio, usted implementará el algoritmo evaluador de expresiones posfijas.

   Escriba una clase *ConvertidorInfijoPosfijo* para convertir expresiones aritméticas con enteros de un solo dígito como

   (6 + 2) * 5 – 8 / 4

a una expresión posfija. La versión posfija de la expresión anterior es

6 2 + 5 * 8 4 / -

La aplicación debe pedir al usuario una cadena infija y luego utilizar una pila para crear la expresión posfija.

El algoritmo para crear una expresión posfija es el siguiente:

a) Meter un paréntesis izquierdo '(' a la pila.
b) Agregar un paréntesis derecho al final de la expresión infija.
c) Mientras la pila no esté vacía, leer la cadena infija de izquierda a derecha carácter por carácter y hacer lo siguiente:

   Si el carácter actual en la expresión infija es un dígito, agréguelo a la expresión posfija.
   Si el carácter actual en la expresión infija es un paréntesis izquierdo, métalo a la pila.
   Si el carácter actual en la expresión infija es un operador:
   i. Saque los operadores (si los hay) en el tope de la pila mientras tengan igual o mayor precedencia que el operador actual y luego agréguelos a la expresión posfija.
   ii. Meta el operador actual a la pila.

   Si el carácter actual en la expresión infija es un paréntesis derecho:
   i. Saque los operadores del tope de la pila y agréguelos a la expresión posfija hasta que se encuentre un paréntesis izquierdo en el tope de la pila.
   ii. Saque y descarte ese paréntesis izquierdo de la pila.

En la expresión se permiten los siguientes operadores aritméticos:

+	suma
-	resta
*	multiplicación
/	división
^	exponenciación
%	módulo

La clase debe tener los siguientes métodos:

a) *ConvertirInfijoPosfijo(string infijo)*: Convierte una expresión infija a posfija, la cual debe retornar.
b) *EsOperador(char operador)*: Si el carácter enviado es un operador devuelve verdadero, en caso contrario devuelve falso.
c) *Precedencia(char operador1, char operador2)*: Devuelve verdadero si el operador1 tiene menor o igual precedencia que el operador2. En caso contrario devuelve falso.

7. Escriba la clase *EvaluadorPosfijo* la cual sirva para evaluar expresiones posfijas tales como

   6 2 + 5 * 8 4 / -

   La aplicación debe recibir una expresión posfija y utilizar una pila para evaluarla.

   El algoritmo para evaluar la expresión posfija es el siguiente:

   a) Agregar un paréntesis derecho ')' al final de la expresión posfija. Cuando se encuentre ese paréntesis , el procesamiento habrá terminado.
   b) Mientras no se encuentre el paréntesis derecho, lea la expresión de izquierda a derecha carácter por carácter:
      Si el carácter actual es un dígito:
         Meter su valor de número entero a la pila.
      Sino, si el carácter actual es un operador:
         Saque del tope de la pila dos elementos y asígnelos a las variables a y b respectivamente.
         Calcule la operación *b operador a*.
         Meta el resultado de la operación en la pila.

   c) Cuando se encuentre el paréntesis derecho en la expresión posfija, saque el valor que esté en el tope de la pila. Ese es el resultado de la evaluación de la expresión posfija.

   La clase debe tener los siguientes métodos:

   a. *EvaluarExpresionPosfija(string posfijo)*: Evalúa la expresión posfija y retorna el resultado.
   b. *Calcular(int operando1, int operando2, char operador)*: Calcula el resultado de *operando1 operador operando2* y retorna el resultado.