Taller de Pilas
Conversión de Números

Objetivo: Implementar un programa informático que, mediante la presentación de un Menú, permita convertir un número en notación decimal (base 10) a su equivalente en sistemas numéricos de base 2 (binario), 8 (octal) o 16 (hexadecimal) utilizando pilas. Utilizar la técnica de divisiones sucesivas para realizar la conversión, almacenando los residuos en una pila estática mediante la función de inserción (Push). La interfaz de usuario debe considerar validaciones de entrada y mensajes de error, también se debe mostrar en pantalla lo que va sucediendo en la pila.

Contextualización: El proceso de la conversión se realiza por medio de una división sucesiva entre la base a la cual el número decimal se está convirtiendo, donde el nuevo divisor es el cociente obtenido de la división anterior. Esta división se repite hasta que el cociente de la división sea 0. El número en la nueva notación se obtiene tomando en orden inverso los residuos obtenidos en la división sucesiva.

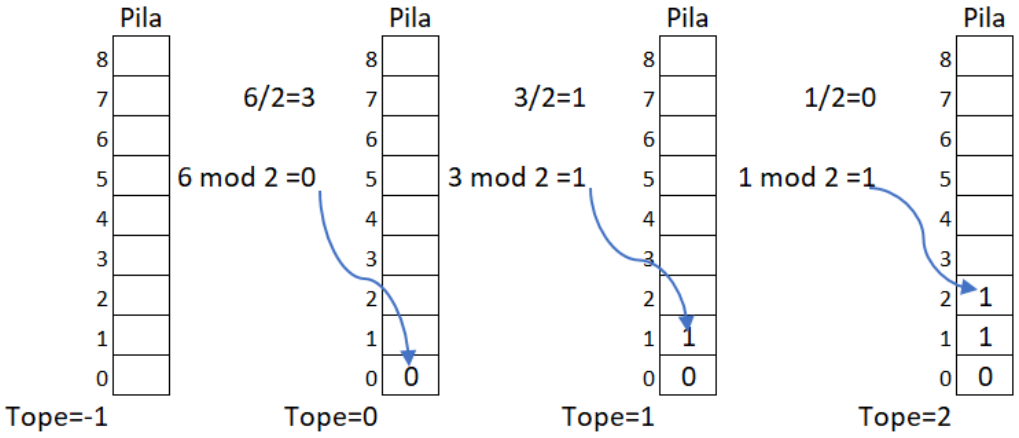
Por ejemplo, al convertir a binario (base 2) el número en notación decimal 6, se requiere de tres divisiones de este tipo:

6/2 =3 residuo 0

3/2 = 1 residuo 1

1/2 = 0 residuo 1

Los residuos 0, 1, 1 se van almacenando uno a uno en una pila estática con el uso de la **función de inserción Push** estudiada en clase, de modo que al recorrer la pila (vector) el equivalente binario de 6 es igual a 110.



Cada vez que se realice una conversión, verificar si la pila está vacía utilizando la función **pilavacía** estudiada en clases. Si no está vacía, eliminar sus elementos con la función de eliminación (Pop).

Nota: Tener en cuenta que al convertir un número en notación decimal (base 10) a su equivalente en un sistema numérico cuya base (radix) es 16 (Hexadecimal), si el residuo almacenado en la pila es mayor o igual a 10, este debe convertirse a su respectiva letra (A, B, C, D, E, F) antes de mostrarlo.

Implementar pruebas unitarias para asegurar el correcto funcionamiento del código y compartir los resultados obtenidos.

Análisis de Complejidad:
Realizar un análisis de complejidad temporal y espacial del algoritmo implementado. Responder las siguientes preguntas:

- ¿Cuál es la complejidad temporal del algoritmo que has implementado para convertir un número decimal a binario utilizando una pila? Explica por qué
- ¿Qué espacio adicional utiliza tu algoritmo para almacenar los residuos en la pila? ¿Cómo afecta esto la complejidad espacial del programa?
- Si decidieras implementar la conversión sin utilizar una pila, ¿cómo cambiaría la complejidad temporal y espacial? Justifica tu respuesta
- Compara la complejidad de tu algoritmo de conversión con otros algoritmos que realicen tareas similares (por ejemplo, conversión entre bases sin pilas). ¿Qué ventajas o desventajas encuentras?

Este trabajo se realizará en grupo de dos (2) estudiantes y subir al aula en el espacio asignado por el docente.

Rubrica de Evaluación

Criterios de Evaluación	Deficiente (1)	Regular (2)	Aceptable (3)	Bueno (4)	Excelente (5)
Funcionalidad del Programa	El programa no funciona o no se presenta.	El programa no cumple con varios requisitos básicos.	El programa tiene varias fallas, pero realiza algunas conversiones.	El programa funciona con pequeñas fallas, pero cumple la mayoría.	El programa funciona perfectamente y cumple con todos los requisitos.
Estructura del Código	Código muy desorganizado y sin comentarios.	Código desorganizado y difícil de seguir.	Código algo desorganizado, con pocos comentarios.	Código organizado y legible, con comentarios en la mayoría.	Código bien organizado, legible y con comentarios claros en todas las funciones.
Análisis de Complejidad	No se presenta análisis de complejidad.	Análisis incompleto o confuso.	Análisis básico, con errores menores.	Análisis correcto, pero con pocos detalles.	Análisis detallado y correcto de la complejidad temporal y espacial.
Interfaz de Usuario	Sin interfaz o interfaz completamente inusable.	Interfaz difícil de usar y poco atractiva.	Interfaz confusa, pero permite el uso del programa.	Interfaz funcional, pero con algunos problemas de usabilidad.	Interfaz intuitiva, amigable y bien diseñada.
Documentación	Sin documentación o incompleta.	Documentación escasa y poco útil.	Documentación básica, con información limitada.	Documentación adecuada, pero podría ser más detallada.	Documentación completa y clara que facilita la comprensión del programa.
Pruebas y Validaciones	No se realizaron pruebas o se presentan resultados inválidos.	Pocas pruebas y sin documentación de resultados.	Se realizaron algunas pruebas, con escasa documentación.	Se realizaron pruebas adecuadas, pero no todas están documentadas.	Se realizaron pruebas exhaustivas con resultados documentados.