Test cases

**Stack**

La implementación fue basada en: <https://www.geeksforgeeks.org/python/stack-in-python/>

# Ejemplo de uso

# Crear una instancia de Stack

MyStack = Stack()

# Agreagar elementos al stack

MyStack.push(10)

MyStack.push(20)

MyStack.push(30)

# Eliminar y ver elementos del stack

print(MyStack.pop())  # Output: 30

# Ver el elemento en la cima del stack

print(MyStack.peek()) # Output: 20

# Verificar el tamaño del stack y si está vacío

print(MyStack.size()) # Output: 2

print(MyStack.is\_empty()) # Output: False

# Intentar eliminar un elemento de un stack vacío

MyStack.pop()  # Elimina 20

MyStack.pop()  # Elimina 10

print(MyStack.pop())  # Output: El stack está vacío \n -1

# Intentar ver el elemento en la cima de un stack vacío

print(MyStack.peek()) # Output: El stack está vacío \n -1

# Verificar el tamaño del stack y si está vacío

print(MyStack.size()) # Output: 0

print(MyStack.is\_empty()) # Output: True

**Queue**

La implementación fue basada en: <https://www.w3schools.com/python/python_dsa_queues.asp>

# Ejemplo de uso

Cola = Queue()

# Agregar elementos a la cola

Cola.enqueue("Perros")

Cola.enqueue("Gatos")

Cola.enqueue("Tortugas")

# Eliminar y ver elementos de la cola

print(Cola.dequeue())  # Output: "Perros"

# Ver el elemento al frente de la cola

print(Cola.peek())    # Output: "Gatos"

# Verificar el tamaño de la cola y si está vacía

print(Cola.size())    # Output: 2

# Verificar si la cola está vacía

print(Cola.is\_empty()) # Output: False

# Intentar eliminar un elemento de una cola vacía

Cola.dequeue()  # Elimina "Gatos"

Cola.dequeue()  # Elimina "Tortugas"

print(Cola.dequeue())  # La cola está vacía, Output: -1

# Verificar si la cola está vacía después de eliminar todos los elementos

print(Cola.is\_empty()) # Output: True

# Intentar ver el elemento al frente de una cola vacía

print(Cola.peek())    # La cola está vacía, Output: -1

# Verificar el tamaño de la cola después de eliminar todos los elementos

print(Cola.size())    # Output: 0

**Hash**

La implementación fue basada en: <https://manishankarjaiswal.medium.com/understanding-hashing-and-hash-tables-in-python-c6336447073d>

# Ejemplo de uso

# Numero de elementos pequeño para forzar colisiones

hash\_table = HashTable(size=5)

hash\_table.insert("nombre", "Juan")

hash\_table.insert("edad", 30)

hash\_table.insert("ciudad", "Madrid")

hash\_table.insert("profesión", "Ingeniero")

hash\_table.insert("hobby", "Fotografía")

hash\_table.insert("deporte", "Fútbol")  # Se agregan 6 para probar colisiones

hash\_table.display()                  # Muestra el contenido de la tabla hash

print(hash\_table.search("nombre"))  # Output: "Juan"

hash\_table.delete("edad")            # Output: "Clave eliminada"

print("Elementos en la tabla hash:", hash\_table.size)             # Output: 5

hash\_table.insert("profesión", "Programador")  # Actualiza el valor

hash\_table.display()                  # Muestra el contenido de la tabla hash