Lab Assignment 14

Please follow the following instructions for Lab Assignment 14

`class Node`

Represents a node in a singly linked list used for separate chaining in the hash table.

- **`__init__(self, key: int) -> None`**
- Initializes a node with a given integer key.
  - **Parameters:**
    - `key` (int): The value to store in the node.
  - **Returns:** `None`
  - **Class Variables:** key, next (initialize using self.key, self.next)

`class HashTable`

Implements a hash table using separate chaining with linked lists.

- **`__init__(self, size: int = 10) -> None`**
  Creates a hash table with the specified number of buckets.
  - **Parameters:**
    - `size` (int): Size of the hash table (default is 10).
  - **Returns:** `None`
  - **Class Variables:** size, table (initialize using self.size, self.table = List containing nodes)

- **`hash_function(self, key: int) -> int`**
  Computes the hash index for a given key using modulo operation. Use key % size. Size should be 10
  - **Parameters:**
    - `key` (int): The key to hash.
  - **Returns:**
    - `int`: The computed index.

- **`insert(self, key: int) -> None`**
  Inserts the key into the hash table using head insertion in the linked list.

- **Parameters:**
  - `key` (int): The key to insert.
- **Returns:** `None`

- **`search(self, key: int) -> bool`**
  Searches for a key in the hash table.
  - **Parameters:**
    - `key` (int): The key to search for.
  - **Returns:**
    - `bool`: `True` if found, otherwise `False`.

- **`delete(self, key: int) -> bool`**
  Deletes a key from the hash table if it exists.
  - **Parameters:**
    - `key` (int): The key to delete.
  - **Returns:**
    - `bool`: `True` if the key was deleted, otherwise `False`.
