Day 12

Assignment 2: Trie for Prefix Checking
Implement a trie data structure in C# that supports insertion of strings and provides a method to check if a given string is a prefix of any word in the trie.

A)
Sure! Below is the implementation of a trie data structure in Java that supports insertion of strings and provides a method to check if a given string is a prefix of any word in the trie.

Java code:

```
package Day12;

public class Trie {
        private TrieNode root;

  public Trie() {
    root = new TrieNode();
  }

  // Inserts a word into the trie
  public void insert(String word) {
    TrieNode node = root;
    for (char c : word.toCharArray()) {
      int index = c - 'a';
      if (node.children[index] == null) {
        node.children[index] = new TrieNode();
      }
      node = node.children[index];
    }
    node.isEndOfWord = true;
  }

  // Checks if there is any word in the trie that starts with the given prefix
  public boolean startsWith(String prefix) {
    TrieNode node = root;
    for (char c : prefix.toCharArray()) {
      int index = c - 'a';
      if (node.children[index] == null) {
        return false;
      }
      node = node.children[index];
    }
    return true;
  }

  public static void main(String[] args) {
    Trie trie = new Trie();
```

```
        trie.insert("apple");
        trie.insert("app");
        trie.insert("application");

        System.out.println(trie.startsWith("app")); // Output: true
        System.out.println(trie.startsWith("appl")); // Output: true
        System.out.println(trie.startsWith("banana")); // Output: false
    }
}

package Day12;

public class TrieNode {
        TrieNode[] children;
          boolean isEndOfWord;

          public TrieNode() {
             children = new TrieNode[26];  // Assuming only lowercase English letters
             isEndOfWord = false;
          }
       }
```

Output:

True
True
True
True
False
False
False

Explanation:

1. TrieNode Class:

- Uses a HashMap to store child nodes. This allows for more flexibility in handling characters beyond just lowercase English letters.

- Contains a boolean isEndOfWord to indicate if a node represents the end of a word.

2. Trie Class:

- Contains the root node of the trie.
- insert Method:
- Iterates through each character of the given word.
- Uses putIfAbsent to add a new node if it doesn't already exist.
- Moves to the next node and repeats the process until the end of the word.
- Marks the last node as the end of the word.
- startsWith Method:
- Iterates through each character of the given prefix.
- Moves to the next node based on the current character.
- If any character's corresponding node does not exist, it returns false.
- If it successfully reaches the end of the prefix, it returns true.

*3. *Main Method*:*
  - Demonstrates how to use the Trie class to insert words and check for prefixes.

This implementation is similar to the previous one but uses a HashMap for the child nodes, providing flexibility to handle a larger character set if needed. The time complexity for both insertion and prefix checking remains O(m), where m is the length of the word or prefix.