

Task 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.

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <stdbool.h>
```

```
#include <string.h>
```

```
#define ALPHABET_SIZE 26
```

```
struct TrieNode {  
    struct TrieNode* children[ALPHABET_SIZE];  
    bool isEndOfWord;  
};
```

```
struct TrieNode* createNode() {  
    struct TrieNode* node = (struct  
TrieNode*)malloc(sizeof(struct TrieNode));
```

```

if (node) {
    node->isEndOfWord = false;
    for (int i = 0; i < ALPHABET_SIZE; i++) {
        node->children[i] = NULL;
    }
}
return node;
}

void insert(struct TrieNode* root, const char* word) {
    struct TrieNode* current = root;
    for (int i = 0; i < strlen(word); i++) {
        int index = word[i] - 'a';
        if (!current->children[index]) {
            current->children[index] = createNode();
        }
        current = current->children[index];
    }
    current->isEndOfWord = true;
}

```

```
bool searchPrefix(struct TrieNode* root, const char*
prefix) {
    struct TrieNode* current = root;
    for (int i = 0; i < strlen(prefix); i++) {
        int index = prefix[i] - 'a';
        if (!current->children[index]) {
            return false;
        }
        current = current->children[index];
    }
    return true;
}
```

```
int main() {
    struct TrieNode* root = createNode();
    insert(root, "apple");
    insert(root, "app");
    insert(root, "banana");
}
```

```

    printf("%s\n", searchPrefix(root, "ap") ? "Prefix
found" : "Prefix not found");

    printf("%s\n", searchPrefix(root, "ban") ? "Prefix
found" : "Prefix not found");

    printf("%s\n", searchPrefix(root, "banan") ? "Prefix
found" : "Prefix not found");

    printf("%s\n", searchPrefix(root, "pear") ? "Prefix
found" : "Prefix not found");

return 0;

}

```

Output: -

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features a header with the Programiz logo and a navigation bar. Below the header, there is a code editor with a file named `main.c`. The code defines a `TrieNode` structure with an array of pointers to other `TrieNode` objects and a boolean flag `isEndOfWord`. It also includes a `createNode` function that allocates memory for a new `TrieNode` and initializes its children array to `NULL`. The `main` function is partially visible, showing the start of the `main` function and the first line of code: `1 #include <stdio.h>`. To the right of the code editor is an `Output` panel. It displays the results of the program's execution: `Prefix found`, `Prefix found`, `Prefix found`, and `Prefix not found`. Below the output, it says `=== Code Execution Successful ===`. The browser's address bar and various icons are visible at the top, and the Windows taskbar is at the bottom.