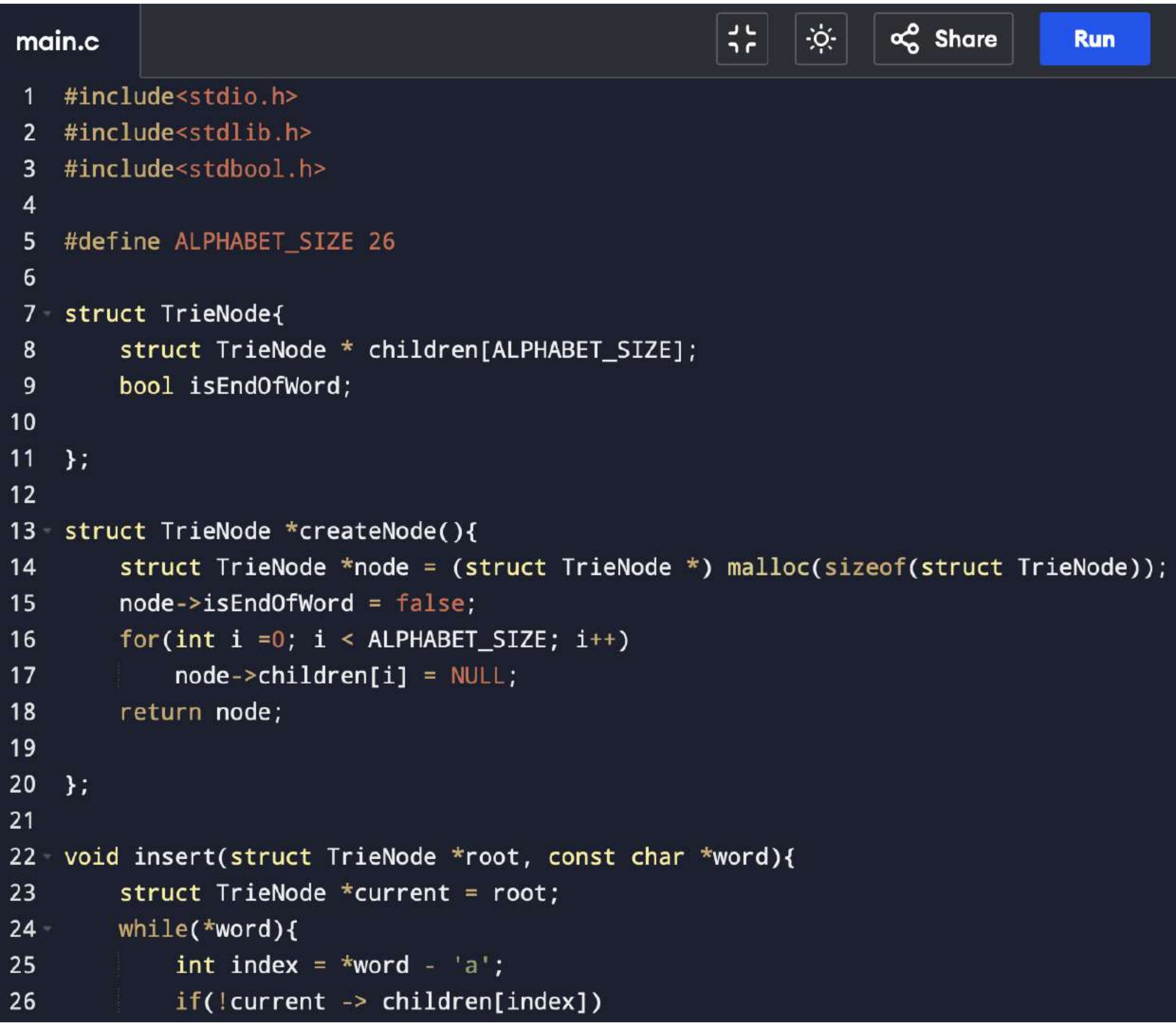
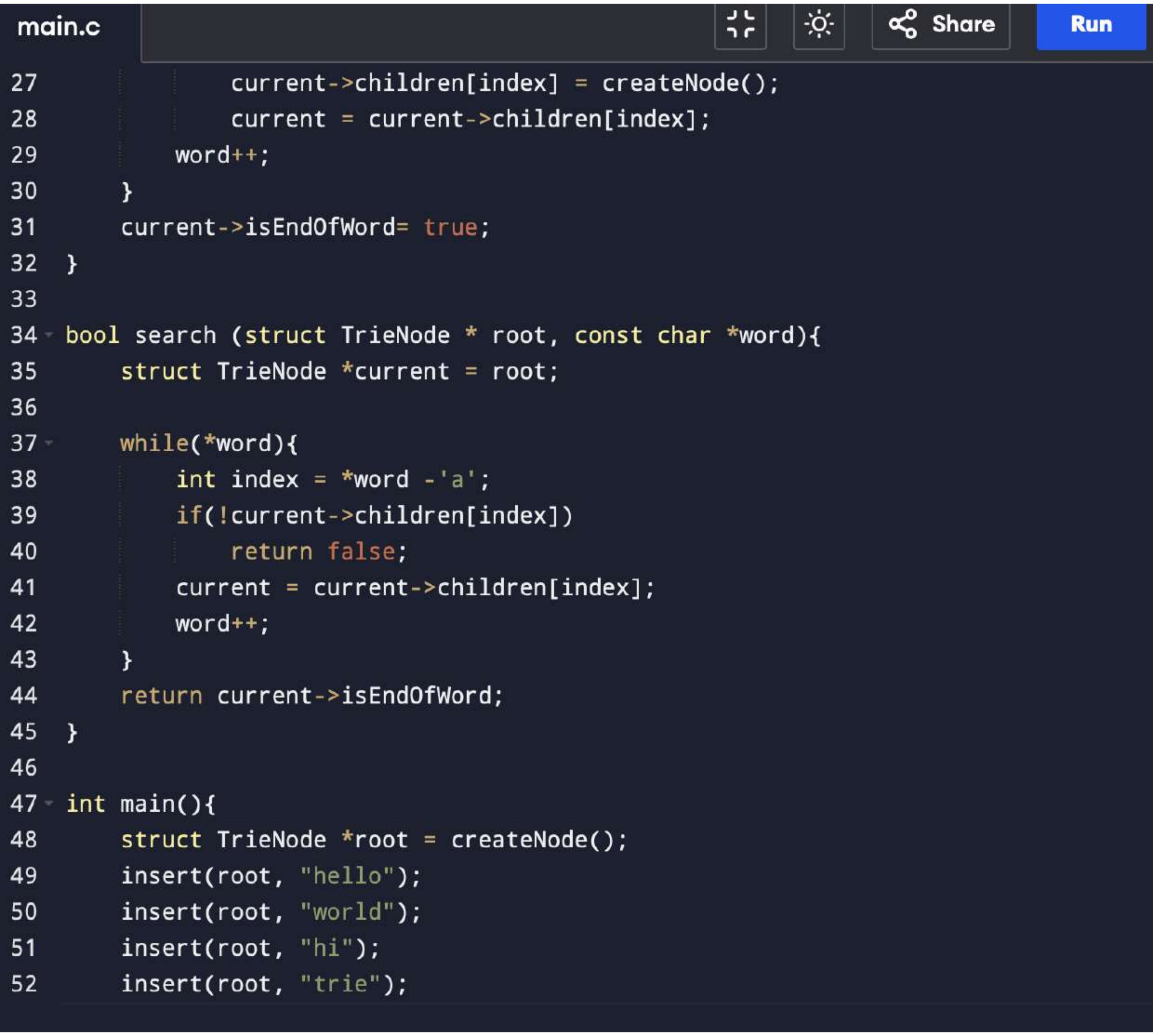
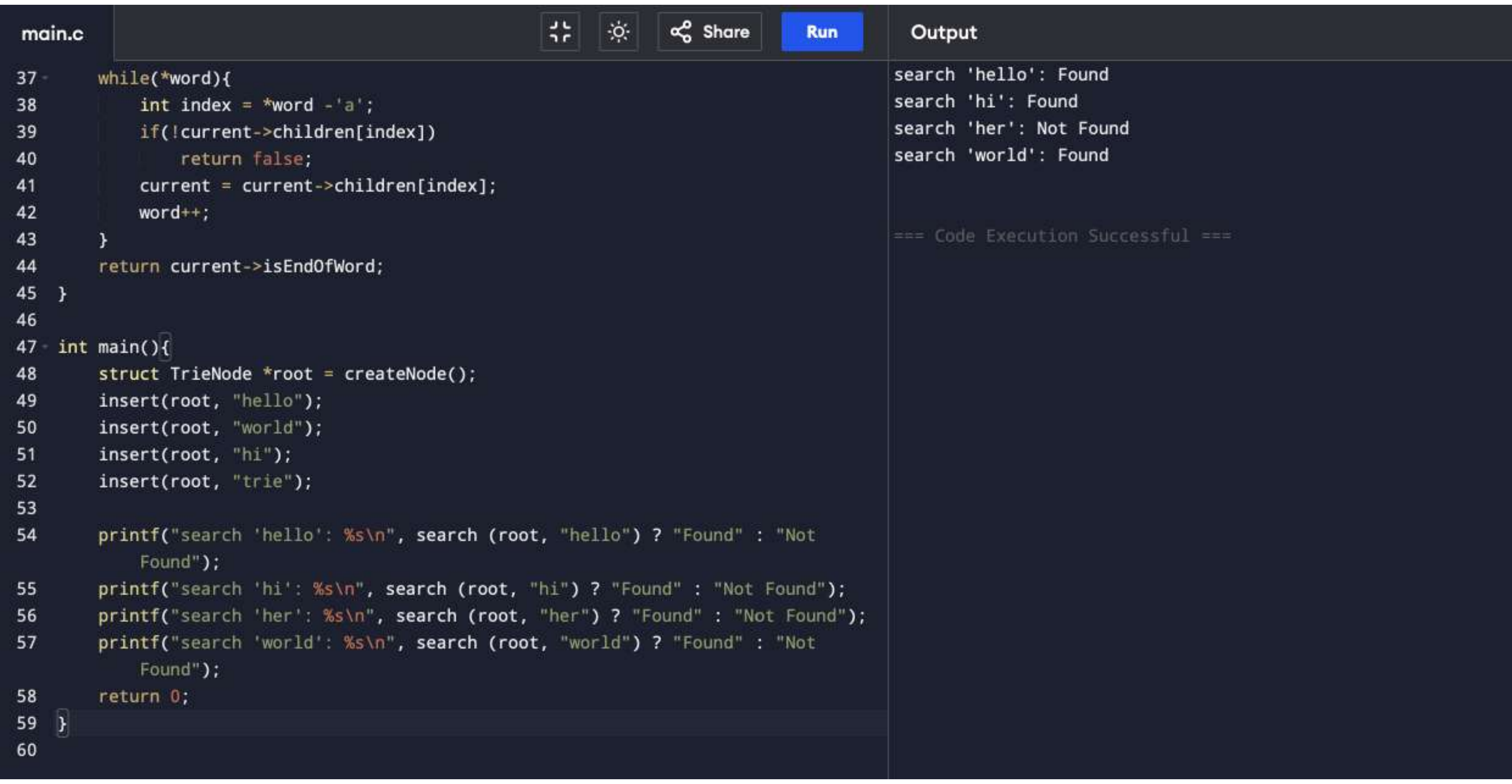


```c
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

#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));
    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;
    while(*word){
        int index = *word - 'a';
        if(!current -> children[index])
```



```c
            current->children[index] = createNode();
            current = current->children[index];
        word++;
    }
    current->isEndOfWord= true;
}

bool search (struct TrieNode * root, const char *word){
    struct TrieNode *current = root;

    while(*word){
        int index = *word -'a';
        if(!current->children[index])
            return false;
        current = current->children[index];
        word++;
    }
    return current->isEndOfWord;
}

int main(){
    struct TrieNode *root = createNode();
    insert(root, "hello");
    insert(root, "world");
    insert(root, "hi");
    insert(root, "trie");
```



```c
    while(*word){
        int index = *word -'a';
        if(!current->children[index])
            return false;
        current = current->children[index];
        word++;
    }
    return current->isEndOfWord;
}

int main(){
    struct TrieNode *root = createNode();
    insert(root, "hello");
    insert(root, "world");
    insert(root, "hi");
    insert(root, "trie");

    printf("search 'hello': %s\n", search (root, "hello") ? "Found" : "Not
        Found");
    printf("search 'hi': %s\n", search (root, "hi") ? "Found" : "Not Found");
    printf("search 'her': %s\n", search (root, "her") ? "Found" : "Not Found");
    printf("search 'world': %s\n", search (root, "world") ? "Found" : "Not
        Found");
    return 0;
}
```

```
search 'hello': Found
search 'hi': Found
search 'her': Not Found
search 'world': Found


=== Code Execution Successful ===
```