S.No: 1	Exp. Name: ***Project Module***	Date:2024-05-09

Aim:

Project Module

Source Code:

hello.c

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

#define ALPHABET_SIZE 26
#define MAX_WORDS 1000
#define MAX_WORD_LENGTH 100

// Trie node structure
typedef struct TrieNode {
    struct TrieNode* children[ALPHABET_SIZE];
    bool isEndOfWord;
} TrieNode;

// Function to create a new Trie node
TrieNode* getNode() {
    TrieNode* node = (TrieNode*)malloc(sizeof(TrieNode));
    node->isEndOfWord = false;
    for (int i = 0; i < ALPHABET_SIZE; i++)
        node->children[i] = NULL;
    return node;
}

// Function to insert a word into the Trie
void insert(TrieNode* root, const char* key) {
    TrieNode* pCrawl = root;
    for (int level = 0; level < strlen(key); level++) {
        int index = key[level] - 'a';
        if (!pCrawl->children[index])
            pCrawl->children[index] = getNode();
        pCrawl = pCrawl->children[index];
    }
    pCrawl->isEndOfWord = true;
}

// Function to search for a word in the Trie
bool search(TrieNode* root, const char* key) {
    TrieNode* pCrawl = root;
    for (int level = 0; level < strlen(key); level++) {
        int index = key[level] - 'a';
        if (!pCrawl->children[index])
            return false;
        pCrawl = pCrawl->children[index];
    }
    return (pCrawl != NULL && pCrawl->isEndOfWord);
```

```
}

// Recursive function to collect all words with a given prefix
void collectWords(TrieNode* root, char* currentPrefix, int level, char results
[MAX_WORDS][MAX_WORD_LENGTH], int* wordCount) {
    if (root->isEndOfWord) {
        currentPrefix[level] = '\0';
        strcpy(results[*wordCount], currentPrefix);
        (*wordCount)++;
    }

    for (int i = 0; i < ALPHABET_SIZE; i++) {
        if (root->children[i]) {
            currentPrefix[level] = 'a' + i;
            collectWords(root->children[i], currentPrefix, level + 1, results,
wordCount);
        }
    }
}

// Function to find completions for a given prefix and return the first comple
tion
void completeAutoWord(TrieNode* root, const char* query, char* completion) {
    TrieNode* pCrawl = root;
    for (int level = 0; level < strlen(query); level++) {
        int index = query[level] - 'a';
        if (!pCrawl->children[index]) {
            printf("No completion found for prefix: %s\n", query);
            strcpy(completion, query);  // If no completion, return the origin
al prefix
            return;
        }
        pCrawl = pCrawl->children[index];
    }

    char currentPrefix[MAX_WORD_LENGTH];
    strcpy(currentPrefix, query);

    char results[MAX_WORDS][MAX_WORD_LENGTH];
    int wordCount = 0;

    collectWords(pCrawl, currentPrefix, strlen(query), results, &wordCount);

    if (wordCount == 0) {
        printf("No completion found for prefix: %s\n", query);
        strcpy(completion, query);  // If no completion, return the original p
refix
    } else {
        strcpy(completion, results[0]);  // Return the first completion
    }
}

int main() {
    TrieNode* root = getNode();

    // Insert a large number of words into the Trie
```

```
    char* keys[] = {
        "the", "a", "there", "answer", "any", "by", "bye", "their",
        "they", "then", "these", "this", "that", "those", "them", "us",
        "up", "upon", "under", "unite", "unity", "use", "user", "usual",
        "utility", "utter", "until", "unique", "universe", "university",
        "umbrella", "ubiquitous", "ultimate", "ultra", "understand",
        "understanding", "undertake", "undermine", "undergo", "underlying", "a
pple", "banana", "app", "ape", "bat", "balloon",
"butterfly", "bakery", "bicycle", "cat", "carrot",
"cabin", "camera", "fat", "family", "fly", "dog",
"doctor", "dollar", "desert", "graphics", "huge",
"items", "mountain", "kitchen", "kit", "keyboard",
"mango", "elephant", "lion", "tiger", "giraffe",
"zebra", "penguin", "dolphin", "whale", "shark",
"octopus", "starfish", "dragonfly", "ladybug", "grasshopper",
"beetle", "antelope", "rhinoceros", "hippopotamus", "crocodile",
"alligator", "snake", "lizard", "frog", "salamander",
"turtle", "tortoise", "kangaroo", "koala", "platypus",
"echidna", "emu", "ostrich", "condor", "baldeagle",
"hawk", "falcon", "raven", "magpie", "robin",
"bluejay", "cardinal", "hummingbird", "woodpecker", "owl",
"parrot", "cockatoo", "canary", "goldfinch", "sparrow",
"crane", "cheetah", "leopard", "panther", "jaguar",
"gazelle", "giraffe", "zebra", "hippo", "rhino",
"elephant", "kangaroo", "koala", "wombat", "dingo",
"kookaburra", "platypus", "emu", "cassowary", "tasmaniandevil",
"crocodile", "alligator", "python", "cobra", "anaconda",
"rattlesnake", "viper", "monitorlizard", "iguan", "gecko",
"chameleon", "salamander", "newt", "frog", "toad",
"turtle", "tortoise", "seaturtle", "terrapin", "boxturtle",
"lion", "tiger", "leopard", "jaguar", "cheetah",
"lynx", "panther", "ocelot", "cougar", "bobcat",
"caracal", "serval", "leopardcat", "cloudedleopard", "fishingcat",
"puma", "wildcat", "marbledcat", "sandcat", "housecat",
"lioness", "tigress", "cub", "calf", "whelp",
"kitten", "cub", "clowder", "pride", "ambush",
"lair", "den", "cage", "zoo", "reserve",
"safari", "savanna", "jungle", "forest", "wood",
"clearing", "grassland", "marsh", "wetland", "delta",
"swamp", "bog", "fen", "quagmire", "mire",
"mud", "muck", "slime", "ooze", "gloop",
"gunk", "mucus", "snot", "phlegm", "sputum",
"spittle", "saliva", "slaver", "drool", "spit",
"gob", "glob", "lugie", "hock", "flem",
"loogie", "mouthful", "sneezeguard", "cough", "splutter",
"sputter", "wheeze", "pant", "gasp", "huff",
"puff", "blow", "exhale", "sigh", "moan",
"growl", "grunt", "snarl", "roar", "bark",
"yelp", "howl", "squeal", "whimper", "whine",
"bawl", "cry", "sob", "sniffle", "snivel",
"blubber", "giggle", "laugh", "chuckle", "snicker",
"titter", "guffaw", "hoot", "snort", "cackle",
"chortle", "giggle", "joke", "jest", "quip",
"pun", "riddle", "limerick", "rhyme", "haiku",
"epic", "ballad", "sonnet", "ode", "elegy",
"lyric", "couplet", "stanza", "verse", "poem",
```

```
"story", "tale", "fable", "legend", "myth",
"epic", "saga", "novel", "novella", "novelette",
"shortstory", "talltale", "tale", "yarn", "fairy tale",
"folk tale", "folklore", "mythology", "myth", "legend",
"story", "tale", "folk tale", "fairy tale", "saga",
"epic", "mythology", "lore", "folklore", "fairytale",
"legends", "folktales", "myths", "epics", "sagas",
"stories", "tales", "fables", "narrative", "narration",
"account", "report", "chronicle", "description", "commentary",
"explanation", "interpretation", "representation","represent","representativ
e", "record", "version","represion","describe","discriptive",
"narrative", "account", "story", "tale", "description",
"chronicle", "report", "commentary", "explanation", "interpretation",
"representation", "record", "version", "history", "chronology",
"record", "chronicle", "annals", "archives", "register",
"documentation", "log", "diary", "journal", "journalism",
"memoirs", "biography", "autobiography", "life story", "life history",
"confession", "testimonial", "documentary", "movie", "film",
"video", "production", "cinema", "cinematography", "motion picture",
"picture", "flick", "feature", "short", "docudrama",
"docufiction", "mockumentary", "animation", "cartoon", "anime",
"manga", "webtoon", "comics", "graphic novel", "sequential art",
"strip", "funnies", "comic strip", "comic book", "pulp",
"penny dreadful", "fanfiction", "slashfiction", "literature", "written works",
"writings", "prose", "fiction", "nonfiction", "writing",
"scripture", "text", "composition", "essay", "novel",
"story", "tale", "poetry", "verse", "drama",
"plays", "scripts", "screenplays", "books", "novels",
"biographies", "autobiographies", "memoirs", "nonfiction", "literature",
"classics", "literary works", "literary classics", "masterpieces", "magnum opu
s",
"literary masterpieces", "opus", "magnus opus", "books", "novels",
"bestsellers", "literary works", "classics", "novellas", "short stories",
"essays", "nonfiction", "memoirs", "biographies", "autobiographies",
"poetry", "verse", "drama", "plays", "screenplays",
"scripts", "works", "written works", "writings", "compositions",
"manuscripts", "scrolls", "folios", "codex", "volumes",
"tomes", "pages", "sheets", "parchments", "papyrus",
"books", "novels", "epics", "sagas", "tales",
"stories", "fables", "folktales", "fairytales", "legends",
"myths", "mythology", "history", "biographies", "autobiographies",
"memoirs", "nonfiction", "essays", "prose", "poetry",
"verse", "drama", "plays", "scripts", "screenplays",
"literature", "written works", "writings", "compositions", "manuscripts",
"scrolls", "folios", "codex", "volumes", "tomes",
"pages", "sheets", "parchments", "papyrus", "newspaper",
"newsmagazine", "journal", "journalism", "broadcast", "broadcasting",
"news", "reporting", "reportage", "coverage", "commentary",
"analysis", "interpretation", "presentation", "exposition", "recitation",
"recital", "reading", "reading aloud", "delivery", "utterance",
"enunciation", "pronunciation", "expression", "articulation", "elocution",
"speech", "talk", "lecture", "address", "sermon",
"harangue", "spiel", "homily", "oration", "peroration",
"apology", "defense", "plea", "petition", "supplication",
"entreaty", "solicitation", "request", "bid", "invitation",
"call"
```

```
        // Add more words as needed...
    };
    int n = sizeof(keys) / sizeof(keys[0]);

    for (int i = 0; i < n; i++)
        insert(root, keys[i]);

    char query[20];
    printf("Enter a prefix to complete: ");
    scanf("%s", query);

    char completion[MAX_WORD_LENGTH];
    completeAutoWord(root, query, completion);
    printf("Completed word: %s\n", completion);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Hello World