Name	: Charles
NIM	: 2802397741
Major	: Computer Science
Course / Class	: COMP6048 – Data Structures / LB01
Lecturer	: Ricky Reynardo Siswanto, S.T., M.Kom.

1. Project Overview

Boogle is a command-line interface (CLI) based slang dictionary application that allows users to manage and explore modern slang terminology. Boogle is developed using the C programming language and uses a trie data structure for efficient storage and retrieval of slang words and their descriptions. Here are some of the basic features that Boogle has:

- Release a New Slang Word
- Search a Slang Word
- View All Slang Words Starting with a Certain Prefix Word
- View All Slang Words

2. Code Snippet and Explanation

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
```

These lines consist of necessary C standard libraries and extra Windows library.

```c
struct TrieNode {
    struct TrieNode *children[128];
    int isEndOfWord;
    char *description;
};
```

This TrieNode structure defines the fundamental of trie data structure for the application:

- children[128] is an array of 128 pointers to other TrieNode(s), representing all possible ASCII characters.
- isEndOfWord is a flag that indicates if this node represents the end of a valid word.
- description is a dynamically allocated string that stores the description of a slang word.

```c
void clearTerminal();
void pressEnterToContinue();
void displayTitle(int titleId);
int isValidSlangWord(char *word);
int isValidSlangDescription(char *description);
int isValidPrefix(char *prefix);
struct TrieNode *releaseMenu(struct TrieNode *root);
void searchMenu(struct TrieNode *root);
void viewMenu(struct TrieNode *root, int isViewPrefix);
struct TrieNode *exitMenu(struct TrieNode *root);
struct TrieNode *createNode();
struct TrieNode *insertWord(struct TrieNode *root, char *word, char *description);
struct TrieNode *searchWord(struct TrieNode *root, char *word);
```

```
void viewWordsWithPrefix(struct TrieNode *root, char *prefix, int *counter);
void viewAllWords(struct TrieNode *root, char *buffer, int depth, int *counter);
struct TrieNode *freeTrie(struct TrieNode *root);
```

These functions declarations define the prototype for all functions used in the application. It allows the compiler to know about the functions before they are fully defined later in the code.

```
int main() {
    struct TrieNode *root = NULL;

    while (1) {
        clearTerminal();
        displayTitle(0);
        printf("\n");
        printf("[1] Release a New Slang Word\n");
        printf("[2] Search a Slang Word\n");
        printf("[3] View All Slang Words Starting with a Certain Prefix Word\n");
        printf("[4] View All Slang Words\n");
        printf("[5] Exit\n");
        printf("\n");

        int userChoice;
        printf("Enter your choice to navigate [1-5]: ");
        scanf("%d", &userChoice); getchar();

        switch (userChoice) {
            case 1:
                root = releaseMenu(root);
                break;
            case 2:
                searchMenu(root);
                break;
            case 3:
                viewMenu(root, 1);
                break;
            case 4:
                viewMenu(root, 0);
                break;
            case 5:
                root = exitMenu(root);
                return 0;
            default:
                printf("Invalid choice, please try again.");
                pressEnterToContinue();
                break;
        }
    }

    return 0;
}
```

The main() function is the entry point of the program. It initializes the root of the trie as null and enters an infinite loop that displays the menu and processes user choices until the user selects the exit option. In this case, enter “1” to release a new slang word, enter “2” to search a slang word, enter “3” to view all released slang words starting with a certain prefix word, enter “4” to view all released slang words without exception, and lastly enter “5” to exit the program which returns 0 when execution completes successfully.

```
void clearTerminal() {
    system("cls"); // system("cls"); to clear the terminal in Windows OS.
}

void pressEnterToContinue() {
    printf("\nPress Enter to Continue...");
    getchar(); // getchar(); to waits for user to press Enter key.
}

void displayTitle(int titleId) {
    switch (titleId) {
        case 0:
            system("chcp 65001 > nul"); // system("chcp 65001 > nul"); to switch to UTF-8 encoding for
box characters.
            printf("██████╗  ██████╗  ██████╗  ██████╗ ██╗     ███████╗\n");
            printf("██╔══██╗██╔═══██╗██╔═══██╗██╔════╝ ██║     ██╔════╝\n");
            printf("██████╔╝██║   ██║██║   ██║██║  ███╗██║     █████╗  \n");
            printf("██╔══██╗██║   ██║██║   ██║██║   ██║██║     ██╔══╝  \n");
            printf("██████╔╝╚██████╔╝╚██████╔╝╚██████╔╝███████╗███████╗\n");
            printf("╚═════╝  ╚═════╝  ╚═════╝  ╚═════╝ ╚══════╝╚══════╝\n");
            system("chcp 437 > nul"); // system("chcp 437 > nul"); to switch back to default encoding.
            break;
        case 1:
            system("chcp 65001 > nul");
            printf("████████╗██╗  ██╗ █████╗ ███╗   ██╗██╗  ██╗    ██╗   ██╗ ██████╗ ██╗   ██╗\n");
            printf("╚══██╔══╝██║  ██║██╔══██╗████╗  ██║██║ ██╔╝    ╚██╗ ██╔╝██╔═══██╗██║   ██║\n");
            printf("   ██║   ███████║███████║██╔██╗ ██║█████╔╝      ╚████╔╝ ██║   ██║██║   ██║\n");
            printf("   ██║   ██╔══██║██╔══██║██║╚██╗██║██╔═██╗       ╚██╔╝  ██║   ██║██║   ██║\n");
            printf("   ██║   ██║  ██║██║  ██║██║ ╚████║██║  ██╗       ██║   ╚██████╔╝╚██████╔╝\n");
            printf("   ██║   ██║  ██║██║  ██║██║ ╚████║██║  ██╗       ██║   ╚██████╔╝╚██████╔╝\n");
            printf("   ╚═╝   ╚═╝  ╚═╝╚═╝  ╚═╝╚═╝  ╚═══╝╚═╝  ╚═╝       ╚═╝    ╚═════╝  ╚═════╝ \n");
            system("chcp 437 > nul");
            break;
        default:
            break;
    }
}
```

- The clearConsole() function clears the screen using the Windows command “cls”.
- The pressEnterToContinue() function waits for the user to press Enter before continuing.
- The displayTitle(titleId) function diplays ASCII art titles, it uses system(“chcp 65001 > nul”) to switch to UTF-8 encoding for proper display of the unicode characters, then switches back to the default encoding after displaying the title.

```
int isValidSlangWord(char *word) {
    // Check length of slang word, reject if the length is less than or equal to 1.
    if (strlen(word) <= 1) {
        printf("Slang word must be more than 1 character.\n");
        return 0;
    }

    // Check for spaces in the slang word, reject if it contains any.
    for (int i = 0; word[i] != '\0'; i++) {
        if (word[i] == ' ') {
            printf("Slang word cannot contain spaces.\n");
            return 0;
        }
    }
    // Check for valid characters (letters, hyphens, underscores), reject if it contains any symbols.
    for (int i = 0; word[i] != '\0'; i++) {
        if (!((word[i] >= 'a' && word[i] <= 'z') || (word[i] >= 'A' && word[i] <= 'Z') || (word[i] == '-
') || (word[i] == '_'))) {
            printf("Slang word should only contain letters, hyphens, or underscores.\n");
            return 0;
        }
    }

    return 1;
}

int isValidSlangDescription(char *description) {
    // Check the description length of slang word, reject if the length is equal to 0 (empty).
    if (strlen(description) == 0) {
        printf("Slang word description cannot be empty.\n");
        return 0;
    }

    // Count the total word count of the description.
    int wordCount = 0, inWord = 0;
    for (int i = 0; description[i] != '\0'; i++) {
        if (description[i] == ' ' || description[i] == '\n') {
            inWord = 0;
        } else if (!inWord) {
            inWord = 1;
            wordCount++;
        }
    }

    // Check the word count of the description, reject if the word count is less than or equal to 2.
    if (wordCount <= 2) {
        printf("Slang word description must contain more than 2 words.\n");
        return 0;
    }

    return 1;
}

int isValidPrefix(char *prefix) {
    // Check the prefix word length, reject if the length is less than or equal to 1 (same as the
minimum length of each slang word).
    if (strlen(prefix) <= 1) {
        printf("Prefix must be more than 1 character.\n");
        return 0;
    }

    // Check for spaces in the prefix word, reject if it contains any.
    for (int i = 0; prefix[i] != '\0'; i++) {
        if (prefix[i] == ' ') {
            printf("Prefix cannot contain spaces.\n");
            return 0;
        }
    }

    return 1;
}
```

- The isValidSlangWord(word) function validates a slang word entered by the user to ensure it meets specific format rules. Firstly, it checks if the word has more than one character, rejecting any that are too short. Then it scans for spaces, which are not allowed in slang words. Finally, it verifies that each character is either a letter (uppercase or lowercase), a hyphen (-), or an underscore (_). If the input fails any of these checks, function returns 0, indicating invalid input. Otherwise, if all conditions are satisfied, it returns 1.

- The isValidSlangDescription(description) function validates the description of a slang word. Firstly, it checks if the description is not empty. Then it counts the number of words in the input by checking space, tab, or newline characters. If the description contains two words or fewer, it is considered too short and the function returns 0. Otherwise, if it contains more than two words, it returns 1.
- The isValidPrefix(prefix) function validates a prefix input to ensure it meets the required format. Firstly, it checks if the prefix has more than one character and contains no spaces. If any of these conditions are not met, the function returns 0. Otherwise, it returns 1.

```
struct TrieNode *releaseMenu(struct TrieNode *root) {
    clearTerminal();
    displayTitle(0);
    printf("\n");

    char word[128], description[256];
    int isValid = 0;

    // Prompt the user to input a new slang word, repeat until the conditions (>1 character, no space)
are met.
    while (!isValid) {
        printf("Enter the new slang word (must be more than 1 character and contains no space): ");
        scanf("%[^\n]", word); getchar();
        isValid = isValidSlangWord(word);
    }

    isValid = 0;

    // Prompt the user to input the description of new slang word, repeat until the condition (>2 words)
is met.
    while (!isValid) {
        printf("Enter the new slang word description (must be more than 2 words): ");
        scanf("%[^\n]", description); getchar();
        isValid = isValidSlangDescription(description);
    }

    // Search for existing slang word; If exist (wordExist = 1), update the description of slang word;
If not exist (wordExist = 0), insert a new slang word.
    struct TrieNode *wordExist = searchWord(root, word);
    if (wordExist) {
        free(wordExist->description);
        wordExist->description = (char *) malloc(strlen(description) + 1); // +1 length for NULL
terminator (\0).
        strcpy(wordExist->description, description);
        printf("\n\"%s\" has been updated!", word);
    } else {
        root = insertWord(root, word, description);
        printf("\n\"%s\" has been released!", word);
    }

    pressEnterToContinue();
    return root;
}
```

The releaseMenu(root) function handles adding new slang words or updating exiting ones. It asks the user to input a word and description, checks if the word already exists, and either insert a new word into the trie or updates the description of an existing word. Firstly, it clears the terminal, displays the title, and asks the user to input a slang word (no spaces, >1 character) and its description (>2 words). Then, it then checks if the word already exits; If found, it frees the old description, allocates memory for the new one, copies it, and print a message that the word was updated; If not found, it inserts the new word and description, then print a message that the word was released.

```
void searchMenu(struct TrieNode *root) {
    clearTerminal();
    displayTitle(0);
    printf("\n");

    char word[128];
    int isValid = 0;

    // Prompt the user to input a slang word, repeat until the conditions (>1
character, no space) are met.
    while (!isValid) {
        printf("Enter the slang word to search (must be more than 1 character and
contains no space): ");
        scanf("%[^\n]", word); getchar();
        isValid = isValidSlangWord(word);
    }

    // Search the requested slang word from user; If found (wordFound = 1), display
slang word and description; If not found (wordFound = 0), display message.
    struct TrieNode *wordFound = searchWord(root, word);
    if (wordFound) {
        printf("\nSlang Word\t: %s\n", word);
        printf("Description\t: %s\n", wordFound->description);
    } else {
        printf("\nThere is no slang word \"%s\" in the dictionary.", word);
    }

    pressEnterToContinue();
}
```

The searchMenu(root) function handles searching for slang words. It asks the user to input a word, uses searchWord(root, word) to find in the trie, and displays the word and its description if found. Firstly, it clears the terminal, displays the title, and asks the user to input a slang word (no spaces, >1 character). Then it calls searchWord(root, word) to search for the word in the trie; If found, it prints the slang word with its description; If not found, it print a message that the slang word doesn’t exist in the dictionary.

```
void viewMenu(struct TrieNode *root, int isViewPrefix) {
    clearTerminal();
    displayTitle(0);
    printf("\n");

    // If isViewPrefix = 1, handle for option [3] View All Slang Words Starting
with a Certain Prefix Word; If isViewPrefix = 0, handle for option [4] View All
Slang Words.
    int counter = 1;
    if (isViewPrefix) {
        char prefix[128];
        int isValid = 0;

        // Prompt the user to input a prefix word, repeat until the conditions (>1
character, no space) are met (same as the condition of each slang word).
        while (!isValid) {
            printf("Enter the prefix word to view all slang words (must be more
than 1 character and contains no space): ");
            scanf("%[^\n]", prefix); getchar();
            isValid = isValidPrefix(prefix);
        }

        printf("\nList of Slang Words with Prefix \"%s\" in The Dictionary:\n",
prefix);
        viewWordsWithPrefix(root, prefix, &counter);
    } else {
        char buffer[128];
        printf("List of All Slang Words in The Dictionary:\n");
        viewAllWords(root, buffer, 0, &counter);
    }

    pressEnterToContinue();
}
```

The viewMenu(root, isViewPrefix) function handles viewing words in the dictionary. It takes a parameter isViewPrefix to determine whether to show all words with specific prefix or all words in the dictionary. Firstly, it clears the terminal and displays the title. It initializes a counter to track and number the displayed words; If isViewPrefix is true, it asks the user to input a prefix word (no spaces, >1 character), then calls viewWordsWithPrefix(root, prefix, counter) to list all slang words starting with that prefix; If isViewPrefix is false, it calls viewAllWords(root, buffer, depth, counter) to display every slang words in the dictionary.

```
struct TrieNode *exitMenu(struct TrieNode *root) {
    root = freeTrie(root);
    clearTerminal();
    displayTitle(1);
    printf("\n");
    return root;
}
```

The exitMenu(root) function handles exiting the program. It frees all memory allocated for the trie using freeTrie(root) function, and displays a thank you message.

```
struct TrieNode *createNode() {
    struct TrieNode *newNode = (struct TrieNode *) malloc(sizeof(struct TrieNode));
    newNode->isEndOfWord = 0;
    newNode->description = NULL;
    for (int i = 0; i < 128; i++) {
        newNode->children[i] = NULL;
    }

    return newNode;
}
```

The createNode() function creates a new trie node. It allocates memory for the node, initializes the isEndOfWord flag to 0, sets the description to null, and initializes all child pointer to null.

```
struct TrieNode *insertWord(struct TrieNode *root, char *word, char *description) {
    if (!root) root = createNode();

    // Traverse the trie and create a new nodes (paths) for characters that don't
exist.
    struct TrieNode *temp = root;
    for (int i = 0; word[i] != '\0'; i++) {
        int arrayIndex = word[i];
        if (!temp->children[arrayIndex]) {
            temp->children[arrayIndex] = createNode();
        }
        temp = temp->children[arrayIndex];
    }

    temp->isEndOfWord = 1; // isEndOfWord = 1 to flag the current node as the end
of a word.
    temp->description = (char *) malloc(strlen(description) + 1); // +1 length for
NULL terminator (\0).
    strcpy(temp->description, description);

    return root;
}
```

The insertWord(root, word, description) function inserts a new word into the trie. It creates the root if it doesn’t exist, traverses the trie following the characters of the word (creating new nodes as needed), marks the final node as the end of a word, and stores its description. Firstly, it check if the root is null; If so, it initializes it by calling createNode(). Then, it uses a pointer temp to traverse the trie character by character based on the input word. For each character, it calculates the index using its ASCII value and checks if a corresponding child node exists; If not, it creates one using createNode(). Finally, it marks the final node with isEndOfWord = 1 to flag the end of a valid word, allocates memory for the description, and copies it.

```
struct TrieNode *searchWord(struct TrieNode *root, char *word) {
    if (!root) return NULL;

    // Traverse the trie by following the path for each character of the word.
    struct TrieNode *temp = root;
    for (int i = 0; word[i] != '\0'; i++) {
        int arrayIndex = word[i];
        if (!temp->children[arrayIndex]) return NULL;
        temp = temp->children[arrayIndex];
    }

    // Return the node if it marks the end of a complete word; otherwise, return
NULL (not found).
    return temp->isEndOfWord ? temp : NULL;
}
```

The searchWord(root, word) function searches for the given word in the trie and returns a pointer to its final node if the word exists. Firstly, it checks if the root is null, returning nothing if the trie is empty. Then, it uses a pointer temp to traverse the trie character by character, calculating each character’s ASCII value to access the appropriate child node. If at any point the required child node doesn't exist, the function returns nothing, indicating the word isn't in the trie. Finally, it checks if the final node marked isEndOfWord = 1; If true, it returns the node; If false, returns nothing.

```
void viewWordsWithPrefix(struct TrieNode *root, char *prefix, int *counter) {
    if (!root) return;

    // Traverse the Trie to reach the node corresponding to the end of the given
prefix word.
    struct TrieNode *temp = root;
    for (int i = 0; prefix[i] != '\0'; i++) {
        int arrayIndex = prefix[i];
        if (!temp->children[arrayIndex] && temp->isEndOfWord) {
            printf("\nThere is no slang word with prefix \"%s\" in the
dictionary.\n", prefix);
            return;
        }
        temp = temp->children[arrayIndex];
    }

    char buffer[128];
    strcpy(buffer, prefix);
    viewAllWords(temp, buffer, strlen(prefix), counter);
}
```

The viewWordsWithPrefix(root, prefix, counter) function displays all words that start with the given prefix. It first navigates to the node representing the end of the prefix, then uses viewAllWords(root, buffer, depth, counter) to display all words from that exact point (depth) in the trie.

```
void viewAllWords(struct TrieNode *root, char *buffer, int depth, int *counter) {
    if (!root) {
        printf("There is no slang word (yet) in the dictionary.\n");
        return;
    };

    if (root->isEndOfWord) {
        buffer[depth] = '\0';
        printf("[%d] %s\n", (*counter)++, buffer);
    }

    for (int i = 0; i < 128; i++) {
        if (root->children[i]) {
            buffer[depth] = i;
            viewAllWords(root->children[i], buffer, depth + 1, counter);
        }
    }
}
```

The viewAllWords(root, buffer, depth, counter) function displays all the words in the trie. It uses a buffer to build words character by character as it traverses the trie, displays complete words when it reaches nodes marked as the end of a word, and uses a counter to number the words.

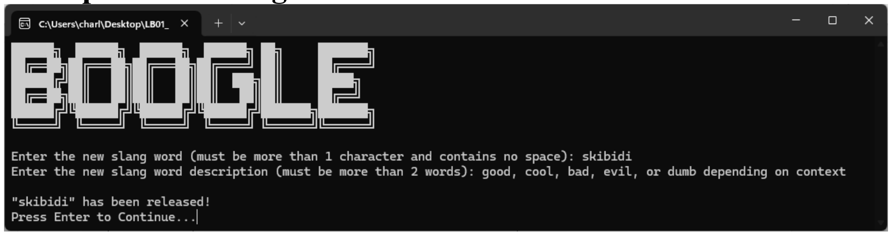
```
struct TrieNode *freeTrie(struct TrieNode *root) {
    if (!root) return NULL;

    for (int i = 0; i < 128; i++) {
        if (root->children[i]) {
            root->children[i] = freeTrie(root->children[i]);
        }
    }

    if (root->description) free(root->description);
    free(root);
    return NULL;
}
```

The freeTrie(root) function frees all memory allocated for the trie. It traverses the trie depth-first, frees all children, frees the description if exists, and frees the node itself. It returns null to indicate that the trie has been freed.

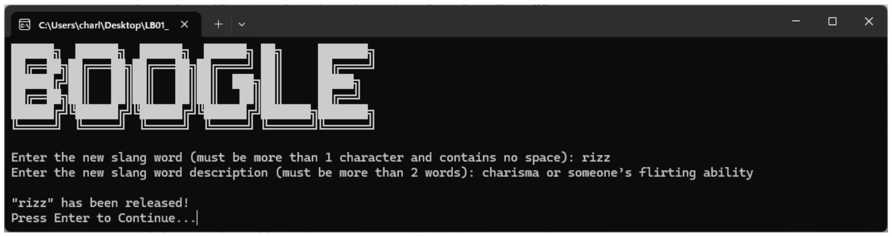
3. Test Cases

Slang Word	**Description**
skibidi	good, cool, bad, evil, or dumb depending on context
rizz	charisma or someone’s flirting ability
cap	not serious or no lie
bussin	excellent, delicious, or impressive
goofy	silly or foolish
delulu	delusional or unrealistic behavior
alpha	social hierarchies among men (leader)
beta	social hierarchies among men (follower)
sigma	social hierarchies among men (independent)
uwu	cuteness or affection
sus	short for suspicious
cringe	embarassing or awkward
blud	friend, buddy, or bro
gyatt	excitement about someone's appearance
mewing	funny facial expression for comedic effect

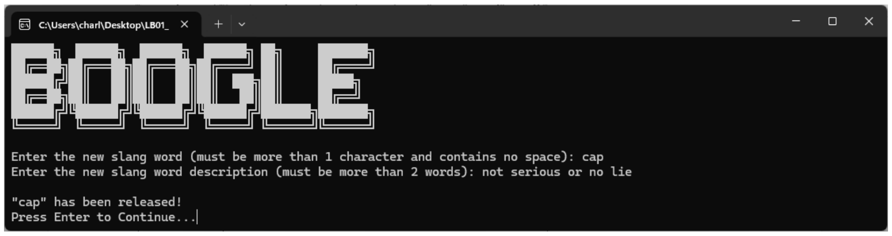
- **Input of 15 Slang Words**



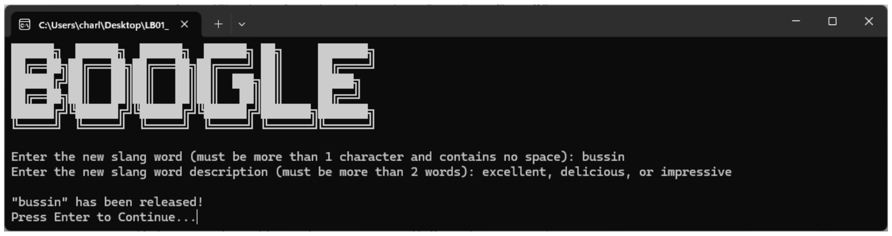
```
Enter the new slang word (must be more than 1 character and contains no space): skibidi
Enter the new slang word description (must be more than 2 words): good, cool, bad, evil, or dumb depending on context

"skibidi" has been released!
Press Enter to Continue...
```



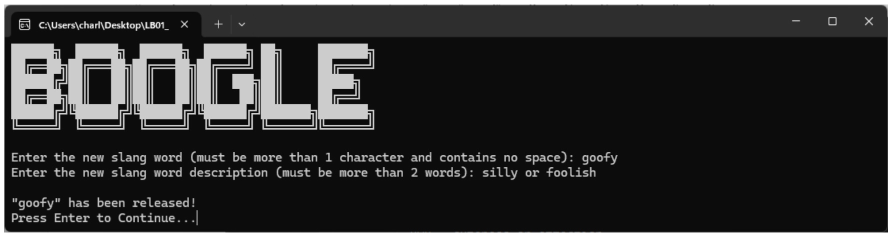
```
Enter the new slang word (must be more than 1 character and contains no space): rizz
Enter the new slang word description (must be more than 2 words): charisma or someone's flirting ability

"rizz" has been released!
Press Enter to Continue...
```



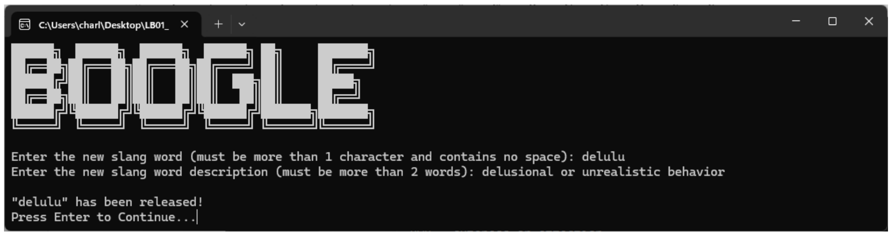
```
Enter the new slang word (must be more than 1 character and contains no space): cap
Enter the new slang word description (must be more than 2 words): not serious or no lie

"cap" has been released!
Press Enter to Continue...
```



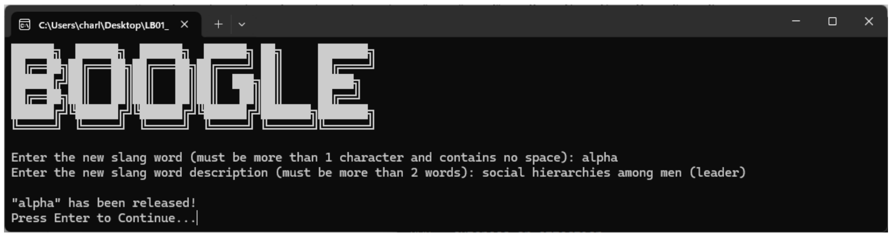
```
Enter the new slang word (must be more than 1 character and contains no space): bussin
Enter the new slang word description (must be more than 2 words): excellent, delicious, or impressive

"bussin" has been released!
Press Enter to Continue...
```



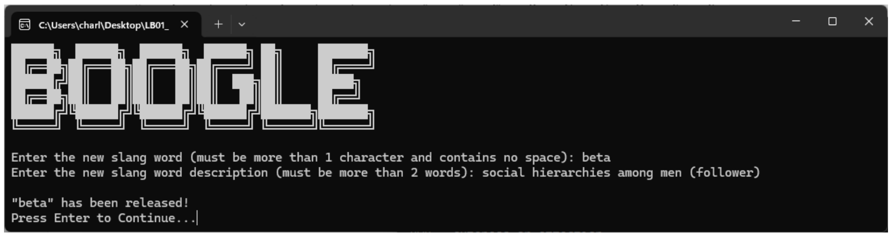
```
Enter the new slang word (must be more than 1 character and contains no space): goofy
Enter the new slang word description (must be more than 2 words): silly or foolish

"goofy" has been released!
Press Enter to Continue...
```



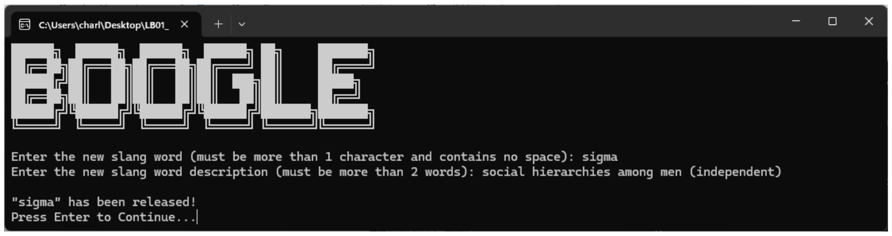
```
Enter the new slang word (must be more than 1 character and contains no space): delulu
Enter the new slang word description (must be more than 2 words): delusional or unrealistic behavior

"delulu" has been released!
Press Enter to Continue...
```



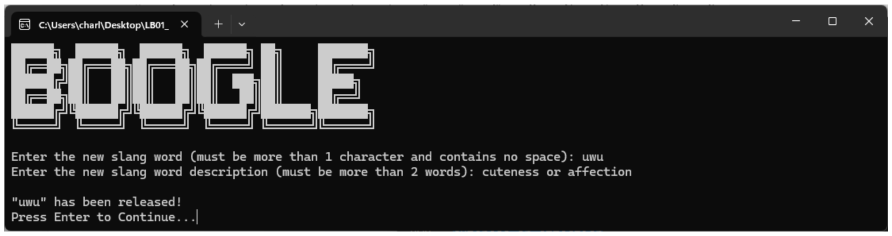
```
Enter the new slang word (must be more than 1 character and contains no space): alpha
Enter the new slang word description (must be more than 2 words): social hierarchies among men (leader)

"alpha" has been released!
Press Enter to Continue...
```



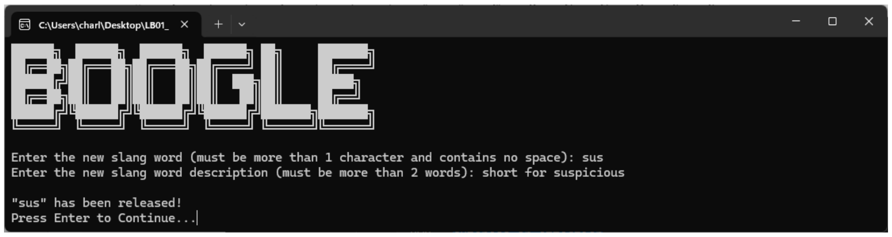
```
Enter the new slang word (must be more than 1 character and contains no space): beta
Enter the new slang word description (must be more than 2 words): social hierarchies among men (follower)

"beta" has been released!
Press Enter to Continue...
```



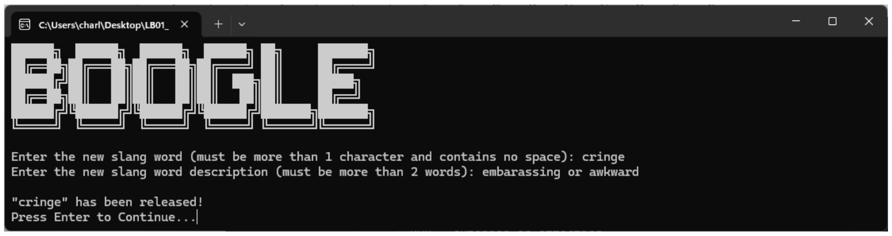
```
Enter the new slang word (must be more than 1 character and contains no space): sigma
Enter the new slang word description (must be more than 2 words): social hierarchies among men (independent)

"sigma" has been released!
Press Enter to Continue...
```



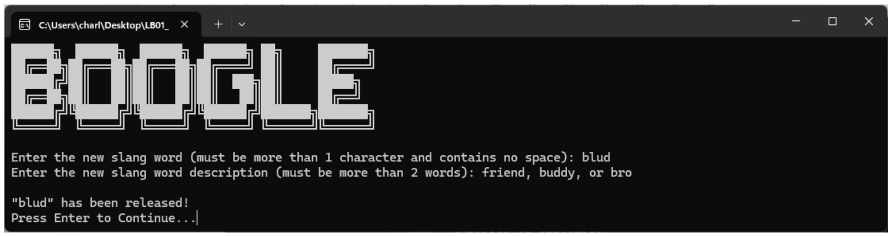
```
Enter the new slang word (must be more than 1 character and contains no space): uwu
Enter the new slang word description (must be more than 2 words): cuteness or affection

"uwu" has been released!
Press Enter to Continue...
```



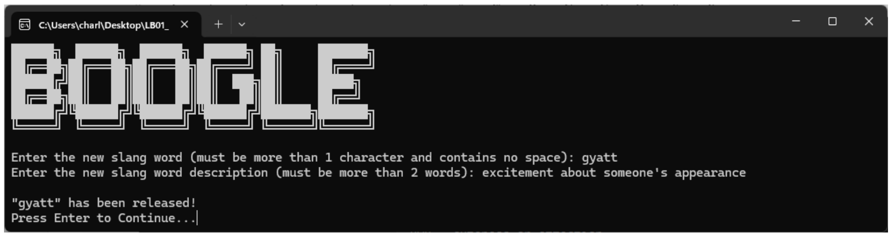
```
Enter the new slang word (must be more than 1 character and contains no space): sus
Enter the new slang word description (must be more than 2 words): short for suspicious

"sus" has been released!
Press Enter to Continue...
```



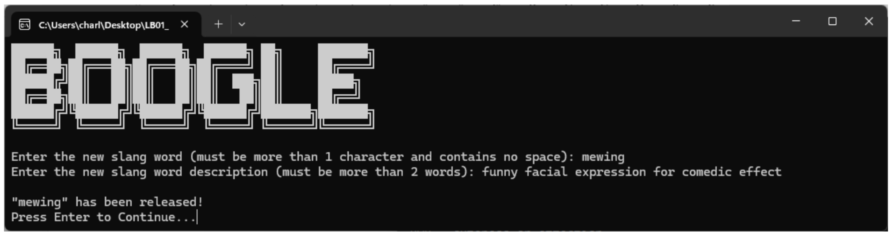
```
Enter the new slang word (must be more than 1 character and contains no space): cringe
Enter the new slang word description (must be more than 2 words): embarassing or awkward

"cringe" has been released!
Press Enter to Continue...
```



```
Enter the new slang word (must be more than 1 character and contains no space): blud
Enter the new slang word description (must be more than 2 words): friend, buddy, or bro

"blud" has been released!
Press Enter to Continue...
```

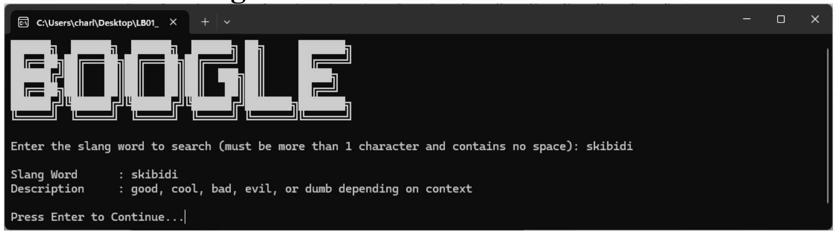


```
C:\Users\charl\Desktop\LB01_
BOOGLE

Enter the new slang word (must be more than 1 character and contains no space): gyatt
Enter the new slang word description (must be more than 2 words): excitement about someone's appearance

"gyatt" has been released!
Press Enter to Continue...
```

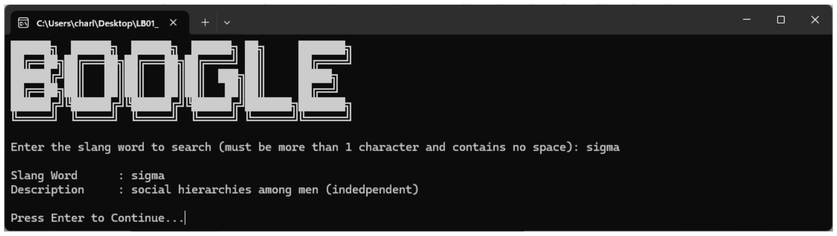


```
C:\Users\charl\Desktop\LB01_
BOOGLE

Enter the new slang word (must be more than 1 character and contains no space): mewing
Enter the new slang word description (must be more than 2 words): funny facial expression for comedic effect

"mewing" has been released!
Press Enter to Continue...
```

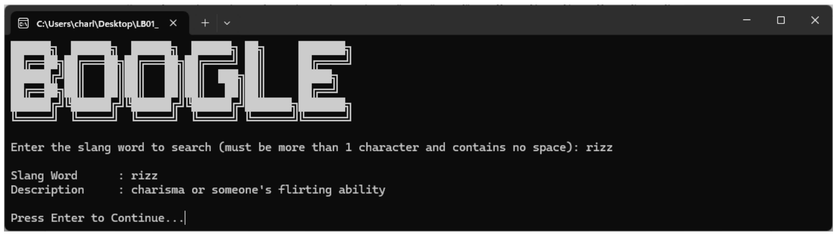
- **Search 5 Slang Words**



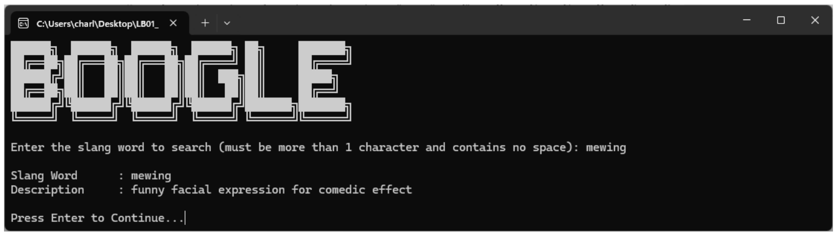
```
C:\Users\charl\Desktop\LB01_
BOOGLE

Enter the slang word to search (must be more than 1 character and contains no space): skibidi

Slang Word      : skibidi
Description     : good, cool, bad, evil, or dumb depending on context

Press Enter to Continue...
```



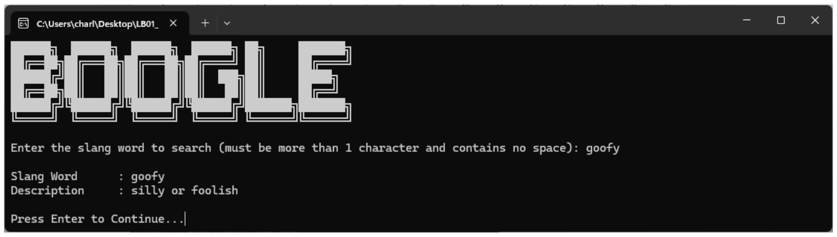
```
C:\Users\charl\Desktop\LB01_
BOOGLE

Enter the slang word to search (must be more than 1 character and contains no space): sigma

Slang Word      : sigma
Description     : social hierarchies among men (indedpendent)

Press Enter to Continue...
```



```
C:\Users\charl\Desktop\LB01_
BOOGLE

Enter the slang word to search (must be more than 1 character and contains no space): rizz

Slang Word      : rizz
Description     : charisma or someone's flirting ability

Press Enter to Continue...
```

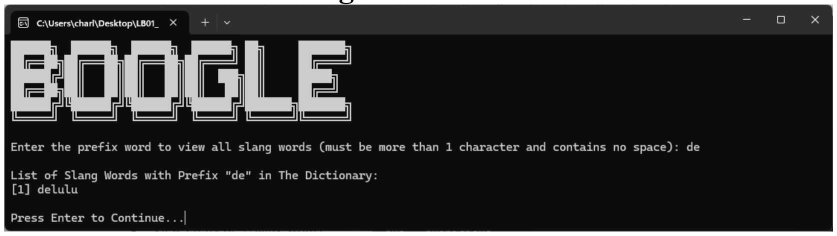


```
C:\Users\charl\Desktop\LB01_
BOOGLE

Enter the slang word to search (must be more than 1 character and contains no space): mewing

Slang Word      : mewing
Description     : funny facial expression for comedic effect

Press Enter to Continue...
```

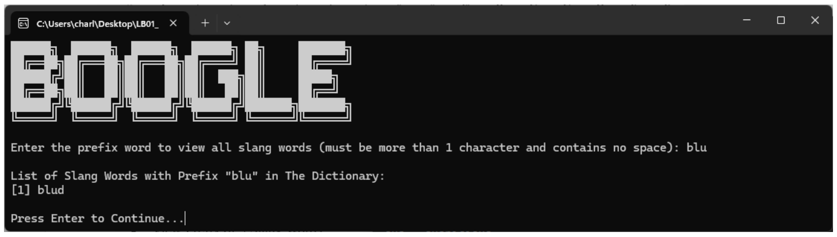


```
C:\Users\charl\Desktop\LB01_

BOOGLE

Enter the slang word to search (must be more than 1 character and contains no space): goofy

Slang Word      : goofy
Description     : silly or foolish

Press Enter to Continue...
```

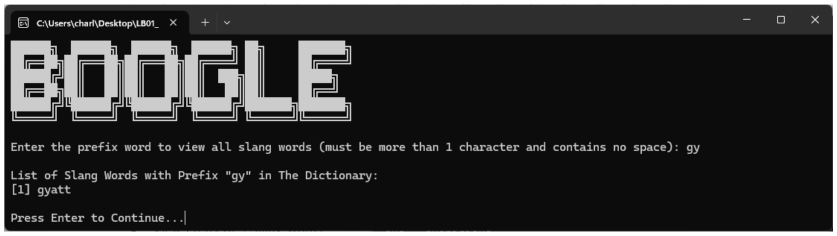
- **View Prefix of 5 Slang Words**



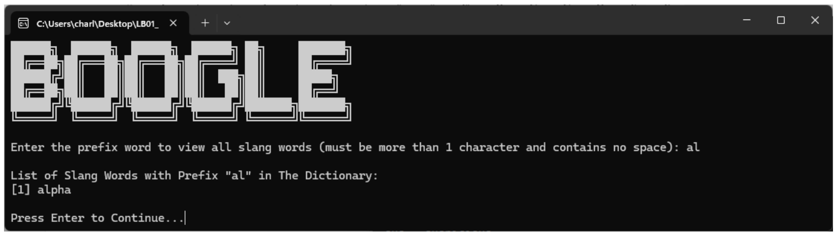
```
C:\Users\charl\Desktop\LB01_

BOOGLE

Enter the prefix word to view all slang words (must be more than 1 character and contains no space): de

List of Slang Words with Prefix "de" in The Dictionary:
[1] delulu

Press Enter to Continue...
```



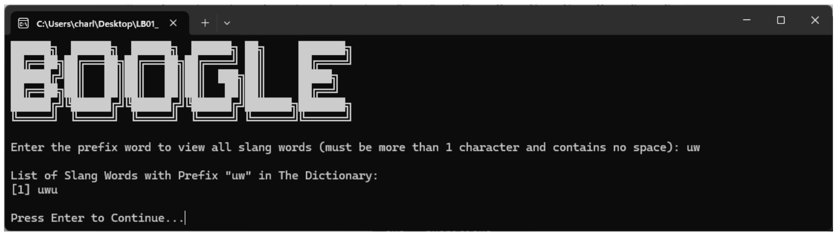
```
C:\Users\charl\Desktop\LB01_

BOOGLE

Enter the prefix word to view all slang words (must be more than 1 character and contains no space): blu

List of Slang Words with Prefix "blu" in The Dictionary:
[1] blud

Press Enter to Continue...
```



```
C:\Users\charl\Desktop\LB01_

BOOGLE

Enter the prefix word to view all slang words (must be more than 1 character and contains no space): gy

List of Slang Words with Prefix "gy" in The Dictionary:
[1] gyatt

Press Enter to Continue...
```

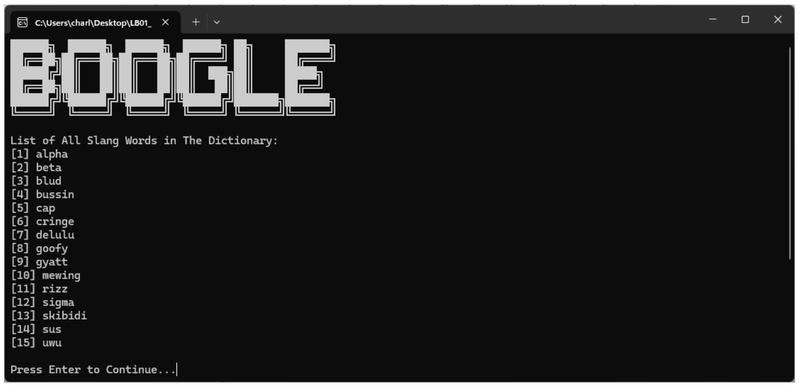


```
C:\Users\charl\Desktop\LB01_

BOOGLE

Enter the prefix word to view all slang words (must be more than 1 character and contains no space): al

List of Slang Words with Prefix "al" in The Dictionary:
[1] alpha

Press Enter to Continue...
```



```
C:\Users\charl\Desktop\LB01_

BOOGLE

Enter the prefix word to view all slang words (must be more than 1 character and contains no space): uw

List of Slang Words with Prefix "uw" in The Dictionary:
[1] uwu

Press Enter to Continue...
```

- **View All Slang Words**



```
BOOGLE

List of All Slang Words in The Dictionary:
[1] alpha
[2] beta
[3] blud
[4] bussin
[5] cap
[6] cringe
[7] delulu
[8] goofy
[9] gyatt
[10] mewing
[11] rizz
[12] sigma
[13] skibidi
[14] sus
[15] uwu

Press Enter to Continue...
```