Boogle adalah aplikasi kamus kata slang. Aplikasi ini memungkinkan pengguna untuk menaan kata slang baru, mencari kata slang yang sudah ada, melihat semua kata slang yang dimulai dengan awalan tertentu, dan melihat semua kata slang yang tersimpan dalam kamus. Program ini menggunakan struktur data Trie untuk menyimpan dan mencari kata-kata dengan efisien.

Logika Penggunaan Trie Dalam Boogle

Boogle menggunakan struktur data Trie untuk menyimpan kata-kata slang. Trie adalah struktur data pohon yang dioptimalkan untuk pencarian string. Setiap node dalam Trie dapat memiliki hingga 26 anak (untuk setiap huruf alphabet a-z). Dalam implementasi ini, setiap node memiliki:

- Array “children[alphabet]” yang menunjuk ke node anak
- Pointer “desc” untuk menyimpan deskripsi kata
- Flag “endOfWord” untuk menandai akhir sebuah kata`

Struktur Trie ini sangat efisien untuk operasi pencarian kata dan pengelompokan kata dengan awalan yang sama.

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

#define alphabet 26        // Jumlah huruf alfabet a-z
#define maxWord 100        // Maksimal panjang kata slang
#define maxDesc 1000       // Maksimal panjang deskripsi slang

// Struktur node pada Trie
typedef struct Node {
    struct Node* children[alphabet];   // Array pointer ke node
anak (a-z)
    char* desc;                        // Deskripsi kata slang
(hanya untuk node akhir kata)
    bool endOfWord;                    // Menandai apakah node
ini akhir sebuah kata
} Node;
```

Penjelasan Fungsi-Fungsi Utama

1. Fungsi createNode()

```
// Membuat node Trie baru
Node* createNode() {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (newNode) {
        for (int i = 0; i < alphabet; i++) {
            newNode->children[i] = NULL;  // Inisialisasi semua
anak NULL
        }
           newNode->desc = NULL;                    // Belum ada
deskripsi
        newNode->endOfWord = false;         // Belum akhir kata
    }
    return newNode;
}
```

Tujuan: Membuat node Trie baru.

Logika: Fungsi ini mengalokasikan memori untuk sebuah node Trie baru, menginisialisasi semua anak node menjadi NULL, dan menandai node tersebut bukan sebagai akhir kata.

Batasan: Jika alokasi memori gagal, fungsi akan mengembalikan NULL.

2. Fungsi validSlang()

```
// Validasi kata slang: harus lebih dari 1 karakter dan tidak
mengandung spasi
bool validSlang(const char* word) {
    if (strlen(word) <= 1) {
        return false;  // Minimal 2 karakter
    }

    for (int i = 0; word[i] != '\0'; i++) {
        if (isspace(word[i])) {
            return false;  // Tidak boleh ada spasi
        }
    }

    return true;
}
```

Tujuan: Memvalidasi format kata slang.

Logika: Fungsi ini memeriksa apakah panjang kata lebih dari 1 karakter dan tidak mengandung spasi.

Batasan: Fungsi ini hanya memeriksa panjang dan keberadaan spasi, tidak memeriksa karakter non-alfabet.

3. Fungsi validDesc()

```
// Validasi deskripsi slang: harus mengandung lebih dari 1 kata
bool validDesc(const char* desc) {
    int wordCount = 0;
    bool inWord = false;

    for (int i = 0; desc[i] != '\0'; i++) {
        if (isspace(desc[i])) {
            if (inWord) {
                inWord = false;
            }
        } else {
            if (!inWord) {
                inWord = true;
                wordCount++;
            }
        }
    }

    return wordCount > 1;  // Harus lebih dari 1 kata
}
```

Tujuan: Memvalidasi format deskripsi kata slang.

Logika: Fungsi ini menghitung jumlah kata dalam deskripsi dan memeriksa apakah jumlahnya lebih dari 1.

Batasan: Fungsi ini menganggap bahwa kata-kata dipisahkan oleh spasi. Tanda baca tidak dihitung sebagai pemisah kata.

4. Fungsi insertWord()

```
// Fungsi memasukkan kata slang dan deskripsinya ke Trie
bool insertWord(Node* root, const char* word, const char* desc)
{
    Node* current = root;
    bool isNewWord = false;

    for (int i = 0; word[i] != '\0'; i++) {
        int index = tolower(word[i]) - 'a';

        // Hanya huruf a-z yang didukung
        if (index < 0 || index >= alphabet) {
            continue;
        }

        if (!current->children[index]) {
              current->children[index] = createNode();   // Buat
node baru jika belum ada
            isNewWord = true;
        }

        current = current->children[index];
    }

      // Jika kata sudah ada, hapus deskripsi lama (update
deskripsi)
    if (current->endOfWord) {
        free(current->desc);
        isNewWord = false;  // Bukan kata baru, cuma update
    }

    // Simpan deskripsi baru dan tandai akhir kata
    current->desc = strdup(desc);
    current->endOfWord = true;

    return isNewWord;
}
```

Tujuan: Menambahkan atau memperbarui kata slang dalam Trie.

Logika: Fungsi ini menelusuri Trie berdasarkan huruf-huruf dalam kata, membuat node baru jika diperlukan. Jika kata sudah ada, deskripsi lama dihapus dan diganti dengan deskripsi baru.

Batasan: Fungsi ini hanya mendukung huruf a-z, dan mengabaikan karakter lain. Program ini menggunakan fungsi strdup() untuk menyalin string deskripsi, yang mengharuskan penggunaan free() untuk membebaskan memori nanti.

5. Fungsi searchWord()

```
// Fungsi mencari kata dalam Trie
Node* searchWord(Node* root, const char* word) {
    Node* current = root;

    for (int i = 0; word[i] != '\0'; i++) {
        int index = tolower(word[i]) - 'a';

        if (index < 0 || index >= alphabet) {
                return NULL;   // Jika ada karakter non-huruf,
langsung gagal
        }

        if (!current->children[index]) {
            return NULL;  // Kata tidak ditemukan
        }

        current = current->children[index];
    }

    if (current && current->endOfWord) {
        return current;  // Ketemu kata dan ini akhir kata
    }

    return NULL;  // Tidak ditemukan
}
```

Tujuan: Mencari kata slang dalam Trie.

Logika: Fungsi ini menelusuri Trie berdasarkan huruf-huruf dalam kata. Jika semua huruf ditemukan dan node terakhir ditandai sebagai akhir kata, fungsi mengembalikan node tersebut.

Batasan: Fungsi ini hanya mendukung huruf a-z, dan mengembalikan NULL jika menemukan karakter lain.

6. Fungsi findAllWithPrefix()

```
// Fungsi rekursif untuk mencari semua kata yang dimulai dengan prefix tertentu
void findAllWithPrefix(Node* root, char* prefix, char* buffer, int depth, char** wordList, int* wordCount) {
    if (!root) {
        return;
    }

    if (root->endOfWord) {
        buffer[depth] = '\0';
           wordList[(*wordCount)] = strdup(buffer);   // Simpan kata ditemukan ke list
        (*wordCount)++;
    }

    for (int i = 0; i < alphabet; i++) {
        if (root->children[i]) {
            buffer[depth] = 'a' + i;
                    findAllWithPrefix(root->children[i], prefix, buffer, depth + 1, wordList, wordCount);
        }
    }
}
```

Tujuan: Mencari semua kata yang dimulai dengan awalan tertentu.

Logika: Fungsi rekursif ini menelusuri Trie mulai dari node yang sesuai dengan awalan. Setiap kali menemukan node yang ditandai sebagai akhir kata, kata tersebut disimpan dalam daftar.

Batasan: Fungsi ini menggunakan rekursi, yang dapat menyebabkan stack overflow jika Trie sangat dalam. Juga, fungsi ini mengasumsikan bahwa wordList sudah dialokasikan dengan ukuran yang cukup.

7. Fungs istoreAllWord()

```
// Fungsi rekursif untuk menyimpan semua kata dari root Trie
void storeAllWord(Node* root, char* buffer, int depth, char** wordList, int* wordCount) {
    if (!root) {
        return;
    }

    if (root->endOfWord) {
        buffer[depth] = '\0';
        wordList[(*wordCount)] = strdup(buffer);
        (*wordCount)++;
    }

    for (int i = 0; i < alphabet; i++) {
        if (root->children[i]) {
            buffer[depth] = 'a' + i;
            storeAllWord(root->children[i], buffer, depth + 1, wordList, wordCount);
        }
    }
}
```

Tujuan: Menyimpan semua kata dalam Trie.

Logika: Fungsi rekursif ini menelusuri seluruh Trie dan menyimpan setiap kata yang ditemukan dalam daftar.

Batasan: Fungsi ini menggunakan rekursi, yang dapat menyebabkan stack overflow jika Trie sangat dalam. Juga, fungsi ini mengasumsikan bahwa wordList sudah dialokasikan dengan ukuran yang cukup.

8. Fungsi displayWordWithPrefix()

```
// Menampilkan semua kata yang diawali prefix tertentu
bool displayWordWithPrefix(Node* root, const char* prefix) {
    Node* current = root;

    // Cari node sesuai prefix
    for (int i = 0; prefix[i] != '\0'; i++) {
        int index = tolower(prefix[i]) - 'a';
        if (index < 0 || index >= alphabet) {
            return false;
        }
```

```
        if (!current->children[index]) {
            return false;
        }
        current = current->children[index];
    }

    // Alokasi list kata sementara
    char** wordList = (char**)malloc(1000 * sizeof(char*));
    int wordCount = 0;

    // Buffer untuk menyimpan kata sementara
    char buffer[maxWord];
    strcpy(buffer, prefix);

    // Cari semua kata dengan prefix
            findAllWithPrefix(current,   (char*)prefix,   buffer,
strlen(prefix), wordList, &wordCount);

    if (wordCount == 0) {
        free(wordList);
        return false;
    }

     // Urutkan kata berdasarkan abjad, seperti urutan kata di
kamus(leksikografis)
    for (int i = 0; i < wordCount - 1; i++) {
        for (int j = i + 1; j < wordCount; j++) {
            if (strcmp(wordList[i], wordList[j]) > 0) {
                char* temp = wordList[i];
                wordList[i] = wordList[j];
                wordList[j] = temp;
            }
        }
    }

    // Tampilkan kata-kata hasil pencarian
    printf("\nWords starts with \"%s\":\n\n", prefix);
    for (int i = 0; i < wordCount; i++) {
        printf("%d. %s\n\n", i + 1, wordList[i]);
    }

    // Bebaskan memori
    for (int i = 0; i < wordCount; i++) {
        free(wordList[i]);
    }
```

```
    free(wordList);

    return true;
}
```

Tujuan: Menampilkan semua kata yang dimulai dengan awalan tertentu.

Logika: Fungsi ini mencari node yang sesuai dengan awalan, kemudian menggunakan findAllWithPrefix() untuk menemukan semua kata. Kata-kata diurutkan secara leksikografis sebelum ditampilkan.

Batasan: Fungsi ini mengalokasikan memori untuk maksimal 1000 kata. Jika jumlah kata lebih dari itu, fungsi tidak akan menampilkan semua kata. Juga, fungsi ini menggunakan selection sort untuk pengurutan, yang tidak efisien untuk daftar kata yang besar.

9. Fungsi displayAllWord()

```
// Menampilkan semua kata slang dalam Trie
bool displayAllWord(Node* root) {
    // Alokasi memori untuk daftar kata
    char** wordList = (char**)malloc(1000 * sizeof(char*));
    int wordCount = 0;

    // Buffer untuk menyimpan kata sementara
    char buffer[maxWord] = {0};

    // Menyimpan semua kata dalam Trie
    storeAllWord(root, buffer, 0, wordList, &wordCount);

    if (wordCount == 0) {
        free(wordList);
        return false;
    }

    // Urutkan kata-kata
    for (int i = 0; i < wordCount - 1; i++) {
        for (int j = i + 1; j < wordCount; j++) {
            if (strcmp(wordList[i], wordList[j]) > 0) {
                char* temp = wordList[i];
                wordList[i] = wordList[j];
                wordList[j] = temp;
            }
        }
    }
```

```
    // Tampilkan semua kata slang
    printf("\nList of all slang words in the dictionary:\n\n");
    for (int i = 0; i < wordCount; i++) {
        printf("%d. %s\n\n", i + 1, wordList[i]);
    }

    // Bebaskan memori
    for (int i = 0; i < wordCount; i++) {
        free(wordList[i]);
    }
    free(wordList);

    return true;
}
```

Tujuan: Menampilkan semua kata dalam Trie.

Logika: Fungsi ini menggunakan storeAllWord() untuk menyimpan semua kata dalam daftar, kemudian mengurutkan dan menampilkan daftar tersebut.

Batasan: Fungsi ini mengalokasikan memori untuk maksimal 1000 kata. Jika jumlah kata lebih dari itu, fungsi tidak akan menampilkan semua kata. Juga, fungsi ini menggunakan selection sort untuk pengurutan, yang tidak efisien untuk daftar kata yang besar.

10. Fungsi freeTrie()

```
// Membebaskan memori semua node Trie
void freeTrie(Node* root) {
    if (!root) {
        return;
    }

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

    if (root->desc) {
        free(root->desc);
    }

    free(root);
```

```
}
```

Tujuan: Membebaskan memori yang digunakan oleh Trie.

Logika: Fungsi rekursif ini menelusuri seluruh Trie dan membebaskan memori yang digunakan oleh setiap node.

Batasan: Fungsi ini menggunakan rekursi, yang dapat menyebabkan stack overflow jika Trie sangat dalam.

11. Fungsi clearInputBuffer()

```
// Membersihkan sisa input pada buffer stdin (contoh saat input
invalid)
void clearInputBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}
```

Tujuan: Membersihkan buffer input.

Logika: Fungsi ini membaca karakter dari buffer input hingga menemukan newline (\n). Ini berguna untuk menghilangkan karakter yang tersisa di buffer input setelah membaca input numerik, untuk mencegah karakter tersebut terbaca sebagai input berikutnya.

Batasan: Fungsi ini hanya membersihkan karakter yang tersisa dalam buffer input saat ini, dan tidak mempengaruhi input yang akan diberikan selanjutnya.

12. Fungsi Utama

```
int main() {
    Node* root = createNode();
    int choice;

    do {
        // Menampilkan menu
              printf("\n=====  Boogle  -  Slang  Word  Dictionary
=====\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("Enter your choice: ");

        if (scanf("%d", &choice) != 1) {
                    printf("\nInvalid input. Please enter a
number.\n");
            clearInputBuffer();
            continue;
        }
        clearInputBuffer();

        char word[maxWord];
        char desc[maxDesc];
        Node* searchResult;

        switch (choice) {
            case 1: // Menambahkan kata slang baru
                do {
                      printf("\nInput a new slang word [Must be
more than 1 characters and contains no space]: ");
                    fgets(word, maxWord, stdin);
                          word[strcspn(word, "\n")] = '\0'; //
Menghapus newline
                } while (!validSlang(word));

                do {
                     printf("\nInput a new slang word desc [Must
be more than 2 words]: ");
                    fgets(desc, maxDesc, stdin);
                          desc[strcspn(desc, "\n")] = '\0'; //
Menghapus newline
                } while (!validDesc(desc));

                if (insertWord(root, word, desc)) {
                       printf("\nSuccessfully released new slang
word.\n");
                } else {
                         printf("\nSuccessfully updated a slang
word.\n");
                }

                printf("\nPress enter to continue...");
                getchar();
                break;
```

```
            case 2: // Mencari kata slang
                do {
                     printf("\nInput a slang word to be searched
[Must be more than 1 characters and contains no space]: ");
                    fgets(word, maxWord, stdin);
                          word[strcspn(word, "\n")] = '\0'; //
Menghapus newline
                } while (!validSlang(word));

                searchResult = searchWord(root, word);
                if (searchResult) {
                    printf("\nSlang word : %s\n\n", word);
                                printf("Description  :  %s\n",
searchResult->desc);
                } else {
                       printf("\nThere is no word \"%s\" in the
dictionary.\n", word);
                }

                printf("\nPress enter to continue...");
                getchar();
                break;

                case 3: // Menampilkan semua kata yang dimulai
dengan awalan tertentu
                printf("\nInput a prefix to be searched: ");
                fgets(word, maxWord, stdin);
                  word[strcspn(word, "\n")] = '\0'; // Menghapus
newline

                if (!displayWordWithPrefix(root, word)) {
                      printf("\nThere is no prefix \"%s\" in the
dictionary.\n", word);
                }

                printf("\nPress enter to continue...");
                getchar();
                break;

            case 4: // Menampilkan semua kata slang
                if (!displayAllWord(root)) {
                     printf("\nThere is no slang word yet in the
dictionary.\n");
                }
```

```
                printf("\nPress enter to continue...");
                getchar();
                break;

            case 5: // Keluar dari program
                printf("\nThank you... Have a nice day :)\n");
                break;

            default:
                        printf("\nInvalid choice. Please try
again.\n");
                printf("\nPress enter to continue...");
                getchar();
        }
    } while (choice != 5);

    // Membersihkan memori
    freeTrie(root);

    return 0;
}
```

Tujuan: Fungsi utama program yang mengimplementasikan menu interaktif dan mengontrol alur program.

Logika: Fungsi ini menampilkan menu, menerima input pengguna, dan memanggil fungsi yang sesuai berdasarkan pilihan pengguna. Program berjalan dalam loop hingga pengguna memilih untuk keluar.

Batasan: Fungsi ini menggunakan scanf() untuk membaca input numerik, yang dapat menyebabkan masalah jika pengguna memasukkan input non-numerik. Oleh karena itu, program menggunakan pemeriksaan nilai kembali scanf() dan membersihkan buffer input jika terjadi kesalahan.

Analisis Kompleksitas

1. Kompleksitas Waktu
    - Menambahkan Kata (insertWord): O(m), di mana m adalah panjang kata.
    - Mencari Kata (searchWord): O(m), di mana m adalah panjang kata.
    - Menampilkan Semua Kata dengan Awalan (displayWordWithPrefix): O(n + k log k), di mana n adalah jumlah node dalam Trie dan k adalah jumlah kata yang ditemukan. Ini karena kita perlu menelusuri Trie (O(n)) dan kemudian mengurutkan kata-kata yang ditemukan (O(k log k)).
    - Menampilkan Semua Kata (displayAllWord): O(n + k log k), di mana n adalah jumlah node dalam Trie dan k adalah jumlah kata dalam Trie.

2. Kompleksitas Ruang
    - Trie: O(alphabet * n * m), di mana n adalah jumlah kata dan m adalah panjang rata-rata kata. Dalam kasus terburuk, setiap node dalam Trie memiliki alphabet anak.
    - Daftar Kata: O(k * m), di mana k adalah jumlah kata yang ditampilkan dan m adalah panjang rata-rata kata.

3. Batasan Program
    - Program hanya mendukung huruf a-z (baik huruf kecil maupun huruf besar). Karakter lain seperti angka, tanda baca, dan karakter khusus akan diabaikan.
    - Program membatasi jumlah kata yang dapat ditampilkan hingga 1000 kata.
    - Program mengalokasikan buffer dengan ukuran tetap untuk kata slang (maxWord = 100) dan deskripsi (maxDesc = 1000), yang berarti kata slang atau deskripsi yang lebih panjang akan terpotong.
    - Program menggunakan rekursi untuk menelusuri Trie, yang dapat menyebabkan stack overflow jika Trie sangat dalam.
    - Program menggunakan selection sort untuk mengurutkan daftar kata, yang tidak efisien untuk daftar kata yang besar.

Case

Test Case #1: Masukan 15 kata slang

Input:

1
da
Same meaning as word "the”

1
gonna
Same meaning as "going to"

1
wanna
Same meaning as "want to"

1
lemme
Same meaning as "let me"

1
gotta
Same meaning as "got to" or "have to"

1
ain’t
Same meaning as "is not", "are not", or "has not"

1
kinda
Same meaning as "kind of"

1
outta
Same meaning as "out of"

1

dunno
Same meaning as "don’t know"

1
yo
Same meaning as "hey" or "hello"

1
bruh
Same meaning as "bro", "dude", or "man"

1
wassup
Same meaning as "what’s up"

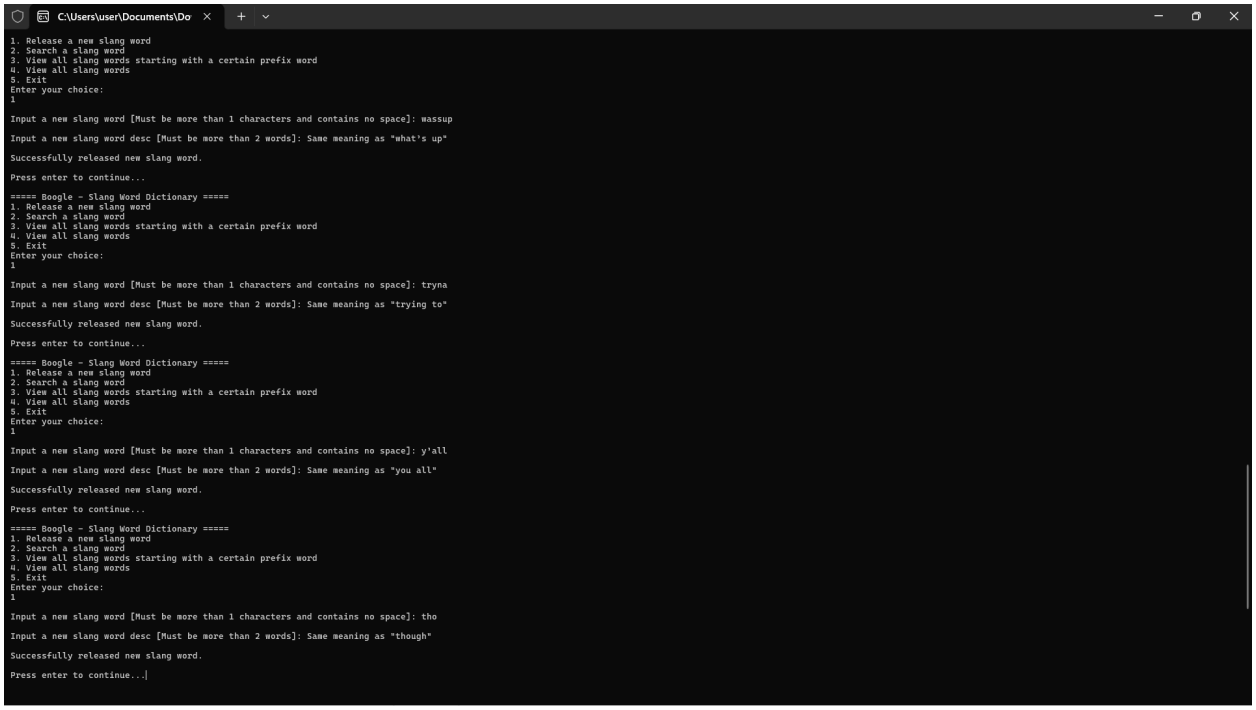
1
tryna
Same meaning as "trying to"

1
y’all
Same meaning as "you all"

1
tho
Same meaning as "though"

Output:

Successfully released new slang word.


C:\Users\user\Documents\Do
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice:
1
Input a new slang word [Must be more than 1 characters and contains no space]: wassup
Input a new slang word desc [Must be more than 2 words]: Same meaning as "what's up"
Successfully released new slang word.
Press enter to continue...
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice:
1
Input a new slang word [Must be more than 1 characters and contains no space]: tryna
Input a new slang word desc [Must be more than 2 words]: Same meaning as "trying to"
Successfully released new slang word.
Press enter to continue...
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice:
1
Input a new slang word [Must be more than 1 characters and contains no space]: y'all
Input a new slang word desc [Must be more than 2 words]: Same meaning as "you all"
Successfully released new slang word.
Press enter to continue...
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice:
1
Input a new slang word [Must be more than 1 characters and contains no space]: tho
Input a new slang word desc [Must be more than 2 words]: Same meaning as "though"
Successfully released new slang word.
Press enter to continue...

Test Case #2: Cari 5 kata

Input:
2
da

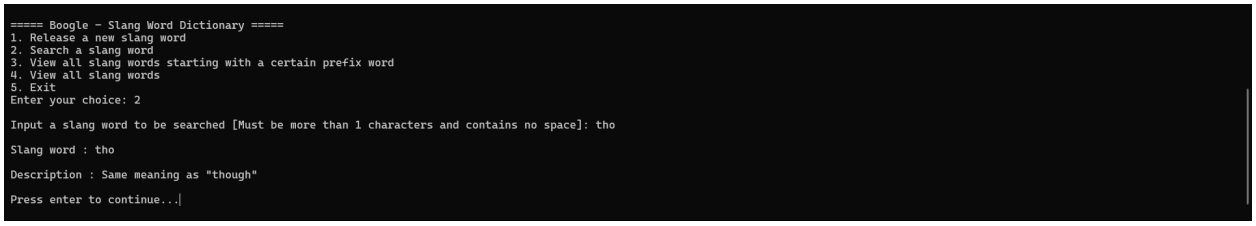
Output:
Slang word : da

Description : Same meaning as word "the"

Input:
2
tho

Output:
Slang word : tho

Description : Same meaning as "though"



Input:
2
dunno

Output:
Slang word : dunno

Description : Same meaning as "don't know"

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 2

Input a slang word to be searched [Must be more than 1 characters and contains no space]: dunno

Slang word : dunno

Description : Same meaning as "don't know"

Press enter to continue...
```

Input:
2
aint

Output:
Slang word : gotta

Description : Same meaning as "got to" or "have to"

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 2

Input a slang word to be searched [Must be more than 1 characters and contains no space]: aint

Slang word : aint

Description : Same meaning as "is not", "are not", or "has not"

Press enter to continue...
```

Input:
2
kentang

Output:
There is no word "kentang" in the dictionary.

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 2

Input a slang word to be searched [Must be more than 1 characters and contains no space]: kentang

There is no Word "kentang" in the dictionary.

Press enter to continue...
```

Test Case 3: Lihat awalan 5 kata

Input:
3
k

Output:
Words starts with "k":

1. kinda

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 3

Input a prefix to be searched: k

Words starts with "k":

1. kinda


Press enter to continue...
```

Input:
3
d

Output:
Words starts with "d":

1. da

2. dunno

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 3

Input a prefix to be searched: d

Words starts with "d":

1. da

2. dunno


Press enter to continue...
```

Input:
3
y

Output:
Words starts with "y":

1. yall

2. yo

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 3

Input a prefix to be searched: y

Words starts with "y":

1. yall

2. yo


Press enter to continue...
```

Input:
3
gon

Output:
Words starts with "gon":

1. gonna

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 3

Input a prefix to be searched: gon

Words starts with "gon":

1. gonna


Press enter to continue...
```

Input:
3
nig

Output:
There is no prefix "nig" in the dictionary.

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 3

Input a prefix to be searched: nig

There is no prefix "nig" in the dictionary.

Press enter to continue...
```

Test Case #4: Lihat semuanya

Input:
4

Output:
List of all slang words in the dictionary:

1. aint
2. bruh
3. da
4. dunno
5. gonna
6. gotta
7. kinda
8. lemme
9. outta
10. tho
11. tryna
12. wanna
13. wassup
14. yall
15. yo

```
===== Boogle - Slang Word Dictionary =====
1. Release a new slang word
2. Search a slang word
3. View all slang words starting with a certain prefix word
4. View all slang words
5. Exit
Enter your choice: 4

List of all slang words in the dictionary:

1. aint

2. bruh

3. da

4. dunno

5. gonna

6. gotta

7. kinda

8. lemme

9. outta

10. tho

11. tryna

12. wanna

13. wassup

14. yall

15. yo


Press enter to continue...
```