Boogle (not google, meme dictionary)

Documentation

Structure:

```
typedef struct TrieNode {
  struct TrieNode *children[ALPHABET_SIZE];
  char *description;
} TrieNode;
```

Boogle menggunakan data structure yang Bernama **Trie** yang konsepnya seperti data structure tree digunakan buat nyimpan "strings" terlebih lagi ini sangat berguna untuk string yang berbagi prefix seperti "dandadan, dana", disini dandadan dan dana sama sama berbagi prefix yang sama yaitu "da", data structure ini juga banyak digunakan di autocomplete, kamus, dan search engines, dan ini juga lebih cepat O(n).

CODE EXPLANATION:

CreateNode()

```
TrieNode *createNode()
```

Tujuan: Untuk membuat node baru dengan bertujuan untuk tempat dimana datanya stringnya akan di simpan

Logic:

Mengalokasikan memory untuk 1 node

```
TrieNode *node = (TrieNode *)malloc(sizeof(TrieNode));
```

Set semua value dari children menjadi NULL

```
  for (int i = 0; i < ALPHABET_SIZE; i++) {
    node->children[i] = NULL; // mengisi children dengan value NULL
  }
```

Set description menjadi NULL juga biar bisa diberikan value yang user mau

```
node->description = NULL;
```

setelah itu kita akan mengembalikan nodenya (return)

```
return node;
```

Limitation: function ini tidak mengecek untuk memory allocation (malloc) yang gagal

FULL CODE:

```
TrieNode *createNode() {
  TrieNode *node = (TrieNode *)malloc(sizeof(TrieNode));
  for (int i = 0; i < ALPHABET_SIZE; i++) {
    node->children[i] = NULL;
  }

  node->description = NULL;
  return node;
}
```

Insert()

```
void insert(TrieNode *root, const char *word, const char *description)
```

Tujuan: untuk memasukkan kata kata baru ke dalam node yang udah user buat beserta deskripsinya
Logic:
Kita akan loop setiap huruf yang user masukkan ke dalam stringnya

```
for (int i = 0; word[i] != '\0'; i++)
```

buat variable ch dan simpan value menjadi huruf kecil

```
char ch = tolower(word[i]);
```

check jika huruf yang kita punya itu huruf ‘a-z’, tapi jika angka kita akan skip ke index selanjutnya

```
if (ch < 'a' || ch > 'z')
    continue;
```

kita akan buat variable index

```
int index = ch - 'a';
```

habis tuh check setiap children yang ada dan check children yang bervalue NULL dan kita akan buat node baru disitu setelah itu kita pindahkan pointernya ke index selanjutnya

```
  if (curr->children[index] == NULL) {
    curr->children[index] = createNode();
  }
  curr = curr->children[index];
```

kita check description kalo descriptionnya itu gak punya value NULL, lebih gampangnya kita replace descriptionnya aja dan mengisinya dengan yang baru

```
 if (curr->description != NULL) {
  free(curr->description);
 }
```

Setelah kita free atau hapus pointer yang mengarah ke memory itu kita isi lagi dengan yang baru dan menreplace semuanya

```
 curr->description = (char *)malloc(strlen(description) + 1);
 strcpy(curr->description, description);
```

limitation: mengabaikan huruf non alphabet (angka, symbol) dan juga overwrite descriptionnya
FULL CODE:

```
void insert(TrieNode *root, const char *word, const char *description) {
 TrieNode *curr = root;
 for (int i = 0; word[i] != '\0'; i++) {
  char ch = tolower(word[i]);
  if (ch < 'a' || ch > 'z')
    continue;
  int index = ch - 'a';
  if (curr->children[index] == NULL) {
    curr->children[index] = createNode();
  }
  curr = curr->children[index];
 }
```

```
  if (curr->description != NULL) {
    free(curr->description);
  }

  curr->description = (char *)malloc(strlen(description) + 1);
  strcpy(curr->description, description);
}
```

Search()

```
void search(TrieNode *root, const char *word)
```

Tujuan: untuk mencari kata kata yang diinginkan user
Logic:
Kita akan loop setiap huruf yang user masukkan ke dalam stringnya

```
for (int i = 0; word[i] != '\0'; i++)
```

buat variable ch dan simpan value menjadi huruf kecil

```
char ch = tolower(word[i]);
```

check jika huruf yang kita punya itu huruf ‘a-z’, tapi jika angka kita akan skip ke index selanjutnya

```
if (ch < 'a' || ch > 'z')
    continue;
```

kita akan buat variable index

```
int index = ch - 'a';
```

habis tuh check setiap children yang ada dan check children yang bervalue NULL dan kita akan kasih output (NOT FOUND, 404) atau di kasus ini karena ini kamus Bahasa gaul jadinya saya masukkan sedikit (banyak) meme

```
  if (curr->children[index] == NULL) {
    printf(
        "The Slang %s word is skibidi and no rizz have no aura, no cap frfr "
        "(not found :v)\n",
        word);
    return;
  }
  curr = curr->children[index];
```

kita check description kalo descriptionnya itu gak punya value NULL, maka kata kata gaulnya ketemu dan kita akan berikan resultnya ke user jika tidak maka kita akan print hal yang sama seperti sebelumnya (NOT FOUND, 404)

```
  if (curr->description != NULL)
    printf("Yapyap for '%s': %s\n", word, curr->description);
  else
    printf("The Slang %s word is skibidi and no rizz have no aura, no cap frfr "
           "(not found :v)\n",
           word);
```

Limitation: case sensitive, gak ada autocomplete ama lambat banget
FULL CODE:

```
void search(TrieNode *root, const char *word) {
  TrieNode *curr = root;
  for (int i = 0; word[i] != '\0'; i++) {
    char ch = tolower(word[i]);
    if (ch < 'a' || ch > 'z')
      continue;

    int index = ch - 'a';
    if (curr->children[index] == NULL) {
      printf(
          "The Slang %s word is skibidi and no rizz have no aura, no cap frfr "
          "(not found :v)\n",
          word);
      return;
    }
    curr = curr->children[index];
  }

  if (curr->description != NULL)
    printf("Yapyap for '%s': %s\n", word, curr->description);
  else
    printf("The Slang %s word is skibidi and no rizz have no aura, no cap frfr "
           "(not found :v)\n",
           word);
}
```

searchByPrefix()

```
void searchByPrefix(TrieNode *root, const char *prefix)
```

Tujuan: Untuk mencari kata kata sesuai dengan prefix (awalan) missal “da” maka output akan “dadadada, dana, daren, dawak, damang, dapa” lengkap dengan deskripsinya
Logic: Masih sama dengan function search() yang beda cuman di bagian akhir functionnya, yang dimana kita copy string dari prefixnya dan masukin ke variable buffer dan print semua value yang kita dapat

```
    strcpy(buffer, prefix);
    printAllWords(curr, buffer, strlen(prefix)); // setelah ini kita jelasin
```

limitation: buffer sizenya gak dynamic dan inputnya gak bisa angka harus huruf a-z
FULL CODE:

```
void searchByPrefix(TrieNode *root, const char *prefix)
{
    TrieNode *curr = root;
```

```
    char buffer[100];

    for (int i = 0; prefix[i] != '\0'; i++)
    {
        char ch = tolower(prefix[i]);
        if (ch < 'a' || ch > 'z')
            continue;

        int index = ch - 'a';
        if (curr->children[index] == NULL)
        {
            printf("hellnahh DaWG We AIn't Got THat WOrd WIht THE '%s' PRefix hel "
                "Nah 💀💀💀💀 (not found :v)\n",
                prefix);
            return;
        }

        curr = curr->children[index];
    }

    strcpy(buffer, prefix);
    printAllWords(curr, buffer, strlen(prefix));
}
```

printAllWords()

```
void printAllWords(TrieNode *node, char *buffer, int depth)
```

Kita bakalan check apa node nya berisi NULL apa nggak, jika iya maka kita hentikan dan langsung kembalikan ke user (return)

```
    if (node == NULL)
        return;
```

Check apa descriptionnya itu bukan NULL jika iya maka kita print resultnya

```
    if (node->description != NULL)
    {
        buffer[depth] = '\0';
        printf("[+100AURA] %s: %s\n", buffer, node->description);
    }
```

Kita check looping dengan total huruf yang ada dan check setiap indexnya apakah dia NULL apa bukan, jika bukan maka kita lakukan recursive

```
    for (int i = 0; i < 26; i++)
    {
        if (node->children[i] != NULL)
        {
```

```
            buffer[depth] = 'a' + i;
            printAllWords(node->children[i], buffer, depth + 1);
        }
    }
```

Limitation: yang ASCII aja yang bisa di print, sedikit lambat
FULL CODE:

```
void printAllWords(TrieNode *node, char *buffer, int depth)
{
    if (node == NULL)
        return;
    if (node->description != NULL)
    {
        buffer[depth] = '\0';
        printf("[+100AURA] %s: %s\n", buffer, node->description);
    }

    for (int i = 0; i < 26; i++)
    {
        if (node->children[i] != NULL)
        {
            buffer[depth] = 'a' + i;
            printAllWords(node->children[i], buffer, depth + 1);
        }
    }
}
```

freeTrie()
Tujuan: menghapus setiap pointer dan membersihkan semua memory yang udah kita pake secara dynamic
Logic:
Jika rootnya NULL maka kita langsung hentikan saja soalnya emang gak ada isi dari memory yang kita tunjuk

```
    if (root == NULL)
        return;
```

kita bakalan lakukan recursive untuk setiap huruf a-z untuk memastikan gak ada anak yang tertinggal

```
    for (int i = 0; i < 26; i++)
    {
        if (root->children[i] != NULL)
        {
            freeTrie(root->children[i]);
        }
```

```
    }
```

Bersihkan juga pointer untuk descriptionnya jika dia memiliki value

```
    if (root->description != NULL)
    {
        free(root->description);
    }
```

Dan lakukan pembersihan total

```
    free(root);
```

Limitation: -

FULL CODE:

```
void freeTrie(TrieNode *root)
{
    if (root == NULL)
        return;

    for (int i = 0; i < 26; i++)
    {
        if (root->children[i] != NULL)
        {
            freeTrie(root->children[i]);
        }
    }

    if (root->description != NULL)
    {
        free(root->description);
    }

    free(root);
}
```

Menu()

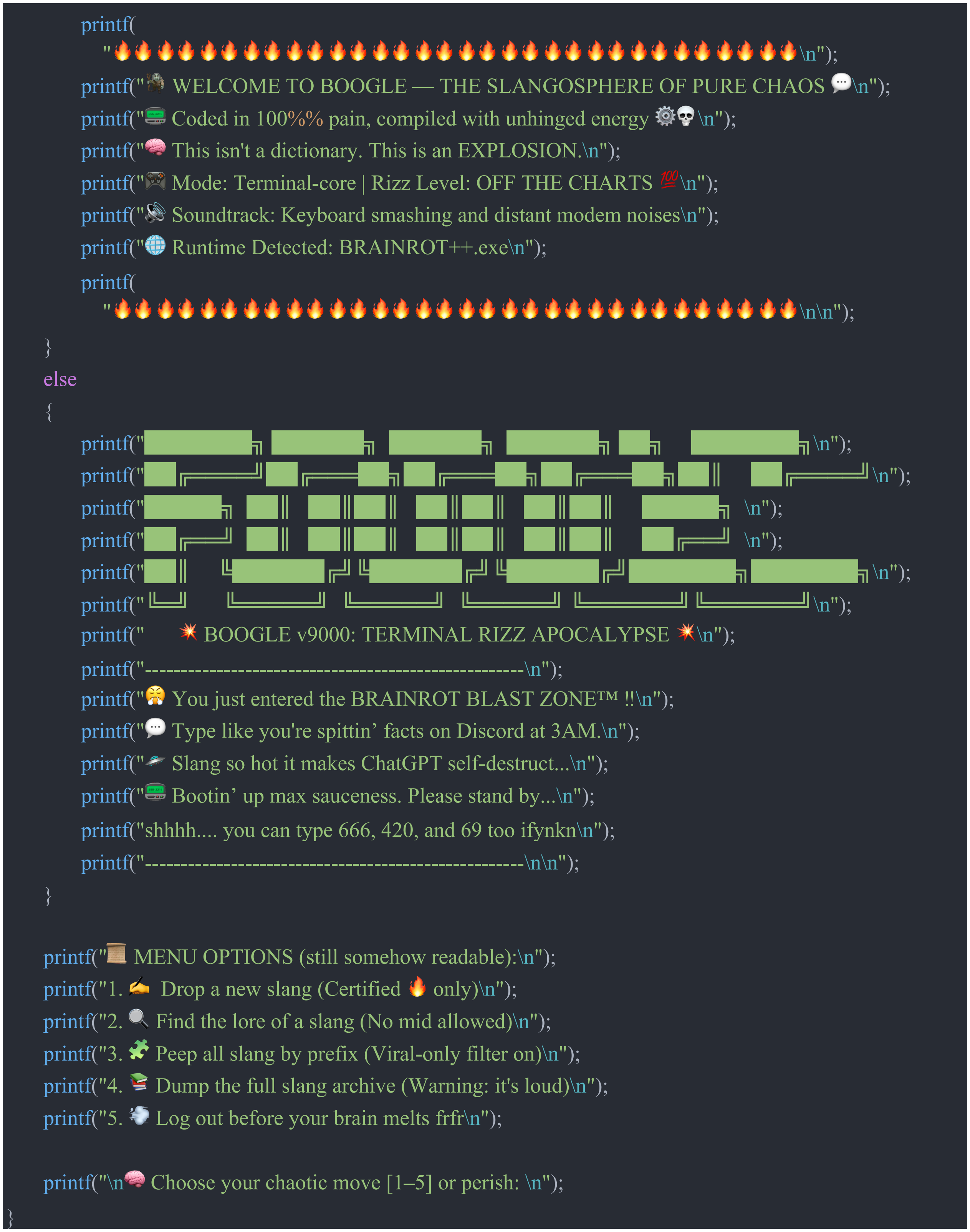
```
void menu()
```

Tujuan: untuk memunculkan menu

Logic: karena disini ada easter egg yang saya masukkan buat user jadi codenya akan sedikit lebih panjang

FULL CODE:

```
void menu()
{
    printf("\n");
    if (easterEGGMENU)
    {
```



```
        printf(
          "🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥\n");
        printf("🧌 WELCOME TO BOOGLE — THE SLANGOSPHERE OF PURE CHAOS 💬\n");
        printf("📟 Coded in 100%% pain, compiled with unhinged energy ⚙️💀\n");
        printf("🧠 This isn't a dictionary. This is an EXPLOSION.\n");
        printf("🎮 Mode: Terminal-core | Rizz Level: OFF THE CHARTS 💯\n");
        printf("🔊 Soundtrack: Keyboard smashing and distant modem noises\n");
        printf("🌐 Runtime Detected: BRAINROT++.exe\n");
        printf(
          "🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥\n\n");
    }
    else
    {
        printf("███████╗ ██████╗  ██████╗  ██████╗ ██╗     ███████╗\n");
        printf("██╔════╝██╔═══██╗██╔═══██╗██╔═══██╗██║     ██╔════╝\n");
        printf("█████╗  ██║   ██║██║   ██║██║   ██║██║     █████╗  \n");
        printf("██╔══╝  ██║   ██║██║   ██║██║   ██║██║     ██╔══╝  \n");
        printf("██║     ╚██████╔╝╚██████╔╝╚██████╔╝███████╗███████╗\n");
        printf("╚═╝      ╚═════╝  ╚═════╝  ╚═════╝ ╚══════╝╚══════╝\n");
        printf("      💥 BOOGLE v9000: TERMINAL RIZZ APOCALYPSE 💥\n");
        printf("--------------------------------------------------\n");
        printf("😤 You just entered the BRAINROT BLAST ZONE™ !!\n");
        printf("💬 Type like you're spittin’ facts on Discord at 3AM.\n");
        printf("🛸 Slang so hot it makes ChatGPT self-destruct...\n");
        printf("📟 Bootin’ up max sauceness. Please stand by...\n");
        printf("shhhh.... you can type 666, 420, and 69 too ifynkn\n");
        printf("--------------------------------------------------\n\n");
    }

    printf("📜 MENU OPTIONS (still somehow readable):\n");
    printf("1. ✍️  Drop a new slang (Certified 🔥 only)\n");
    printf("2. 🔍 Find the lore of a slang (No mid allowed)\n");
    printf("3. 🧩 Peep all slang by prefix (Viral-only filter on)\n");
    printf("4. 📚 Dump the full slang archive (Warning: it's loud)\n");
    printf("5. 💨 Log out before your brain melts frfr\n");

    printf("\n🧠 Choose your chaotic move [1–5] or perish: \n");
}
```

Kobokan()

```
void koboKan()
```

tujuan: gak ada cuman buar have fun aja
logic: just a bunch of printf that makes ASCII ART
FULL CODE:

```
void koboKan()
{
    printf("                    :-:-:::     ::..:::::=                "
        "                       \n");
    printf("                    -======..  =:.:.=---::::                "
        "                       \n");
    printf("                   -:-    ==.-=:.::- ------..                "
        "                       \n");
    printf("            -.=      :::=     =:::-      -=:-..             "
        "                       \n");
    printf("              ..      ::::.-    --:=        -.::..           "
        "                       \n");
    printf("      =...*    :.=     = -:::      =-         -.....         "
        "                       \n");
    printf("        *...=   ::          -::-.....::..     ::.::.         "
        "                       \n");
    printf("          *:::  ::       =:--=:::=+:::::..      :            "
        "                       \n");
    printf("            *:: ::      +:==+*  =:---:: -::..:                "
        "                       \n");
    printf("         *+==+:-::     =:*+:+::-:  :::-:: ::.:-:              "
        "                       \n");
    printf("      *======.:...:   =*=+--=:::--. :::-:: -:::::             "
        "                       \n");
    printf("     *+=+*=.::::::-*:-:-=++-:=:--: -  ..=:-:::===::.:         "
        "                       \n");
    printf("     +=====-::=-:=-=*+::++:-::::-::-  . ::=::::-:..:          "
        "                       \n");
    printf("    *=+==*===-=-*+#=+=+=+-+==:-*:--:: .. .-=::==:...          "
        "                       \n");
    printf("    ++=======:-=*=-=-=+==*+:-=-::::===-.. .==::-:....         "
        "                       \n");
    printf("   ++==++=====+=kobo-=+=+-==:= -:=.=..-.= .-==:::.::.         "
        "      ..               \n");
    printf("   *==+++++=:====-=++-=*-+=.+.*::-:#*.:-:. ..::-:-:.::.       "
        "    .....              \n");
    printf("   +====++++======++ -:+*:=:-:::::::.:=..:. .::*=:=:.::..     "
```

```
        "  ........                    \n");
    printf("    ==============+.:::.:.:*:\%%##:::::.::::.....#+==:=:=:::.    :.:. "
        " :   ::.... ::                 \n");
    printf("    ---==========\%% .  :.:*+==::::.::::::::.-===:::-.===::.:.....  "
        ":.:.::.....:..:  :.            \n");
    printf("     +----=-==*:.  ..  .:**==:::::===::::+=--.=-==-::::-:::..... "
        "-::.:.....    :::              \n");
    printf("       "
        "=--=\%%\%%--::.....:.:...:--=+::::---:::#=:**=+:--:=::=:-+=-:-:--::.....:."
        ".  ....::.:::              \n");
    printf("         "
        "*=:==-==--::.::::..:=++++*::::===-#==*=+-=:--:::+======-:-::.::::::  "
        ".:-..:.....::::::          \n");
    printf("           ==-:====:::::::-:-*+*+++#++==-:-+::**=-:=::=:-.:.====---  "
        "       ::::. :::::.:          \n");
    printf("             *=======:::::-:--*:*-++++:---===*=:---: "
        "::+-:..:====-=--:.::::::.::::::::::::          \n");
    printf("               +-:--::=:::=-*+=++=:#======----:=-::::=   :.:= "
        "=::==---::::--::::::--:.....:          \n");
    printf("                  -=====*-=----*=====+=*=-==:=:=-:: .-:-*  ::    "
        "=::::::::::.::::---:::::            \n");
    printf("                        *+:+:========-===*:--=.:::::::::+  :      "
        ":::.::::::::::::::::.. : :      \n");
    printf("                        *..=..+:::\%%=*::+.::-... :-:  :::-        "
        ":..:  :::-:::.::::::::::::        \n");
    printf("                    * :=...=.=..\%%#:+:..+.#::.::::..  :::-       "
        "\%%**+**+.::::::.::::::          \n");
    printf("                  *     :.  :.....:.+-=:-+::::::::.. ....:=   "
        "\%%****+*+=::=::::  :::.::::  ::       \n");
    printf("                      = :: .#.   .. "
        "+:.....::::::-:::.....::+**+#+**-*--===+ :::   ..::.:::          \n");
    printf("               *     - :::::. \%%   *.   "
        "..==-:=-::.....:+++==\%%++-=*=*+++=*       :::::..:         \n");
    printf("                        :::**: ::: . *    "
        ".::=::=--::-::.-\%%+=====#--:=::===-:::=     :  :   :         \n");
    printf("                        *::+:*.*:.... . ....:::== "
        "=+===:***\%%++===*=+*-+-:=*==+*                    \n");
    printf("                      -*#:::-::-::=::+...:=-.: : "
        "=-=====*##=+=====*==*=:-:+*+=*                    \n");
    printf("                      ==#::-#::-#::::+::.....:: "
        ":===-=:=-#=+======*+*=+*=*+*+                    \n");
```

```
printf("                    "
       "-=#\%%=-*:::---::+::::::::::+===-=:==*=++=+==*=+*#*++*+            "
       "          \n");
printf("                    "
       "=#\%%:-=-:--#=--=::::::::*++:=*=--:--========:=+=+**              "
       "         \n");
printf("                    "
       "\%%\%%:***+::=-:+::::::\%%***:::: =-==#++======+++=*              "
       "          \n");
printf(
    "                    *==-:*******====****#-=--::=:=-#--========* "
    "                    \n");
printf("                    =+-::::#\%%*:**=+++:::-------* "
       "::+*-------*                    \n");
printf("                    =.-*::::#+:=.:::::::=----=:::-::       "
       "                    \n");
printf("                     -=#:::::::-+=:::::::+::--====-        "
       "                    \n");
printf("                      *=#=:::::-:=.:::::::*-==:==-         "
       "                    \n");
printf("                      = =#+:: :::::::.*:::===:==-          "
       "                    \n");
printf("                      +   -::::::==::==-=-===:.=           "
       "                    \n");
printf("                            ::::**==-===-=:::::::=         "
       "                    \n");
printf("                             *-:-.:=  +   *:::::::.        "
       "                    \n");
printf("                               =:=+*  =    =:-::::         "
       "                    \n");
printf("                                 **::  =    +::::*         "
       "                    \n");
printf("                                  *:.       +*#**         "
       "                    \n");
printf("                                   :*       *:::*         "
       "                    \n");
printf("                                  =+*=       :::          "
       "                    \n");
printf("                                   -::==    *-*:+          "
       "                    \n");
printf("                                   =-=*+    *+-*          "
```

```
        "                    \n");
    printf("                          *++  =-*:++       "
        "                    \n");
    printf("                              +=**=        "
        "                    \n");
    printf("                                      "
        "                    \n");
}
```

Limitation: user terminal UI beberapa terminal kayaknya gak bisa load ASCII ART

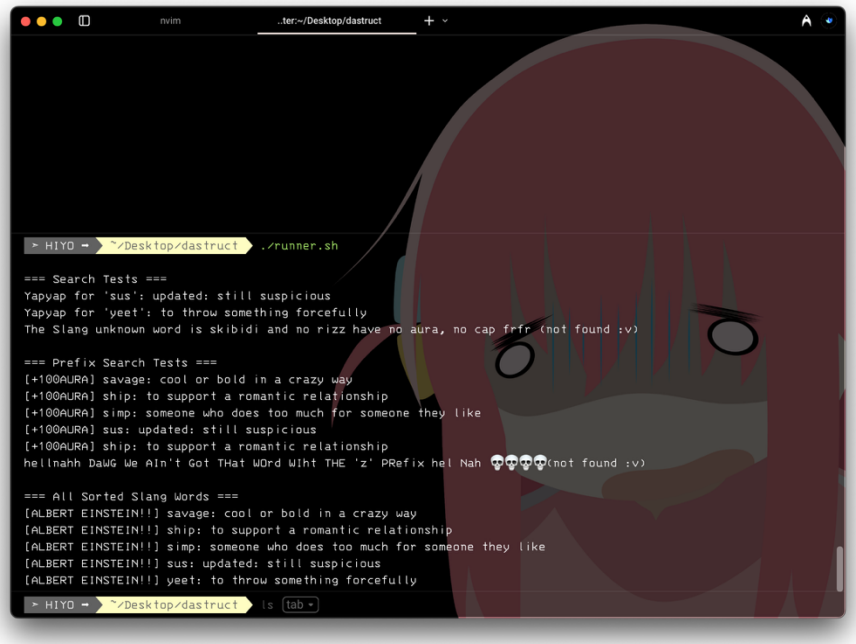
CUSTOM CASE:

```
    insert(root, "sus", "suspicious behavior");
    insert(root, "yeet", "to throw something forcefully");
    insert(root, "simp", "someone who does too much for someone they like");
    insert(root, "sus", "updated: still suspicious"); // overwrite test
    insert(root, "savage", "cool or bold in a crazy way");
    insert(root, "ship", "to support a romantic relationship");

    // 🔎 Search exact
    printf("\n=== Search Tests ===\n");
    search(root, "sus");    // should show updated description
    search(root, "yeet");   // should show original description
    search(root, "unknown"); // should not be found

    printf("\n=== Prefix Search Tests ===\n");
    searchByPrefix(root, "s"); // should list sus, simp, savage, ship
    searchByPrefix(root, "sh"); // should list ship
    searchByPrefix(root, "z"); // should say not found

    // 📚 Print all in order
    printf("\n=== All Sorted Slang Words ===\n");
    char buffer[100];
    printAllWords(root, buffer, 0);
```



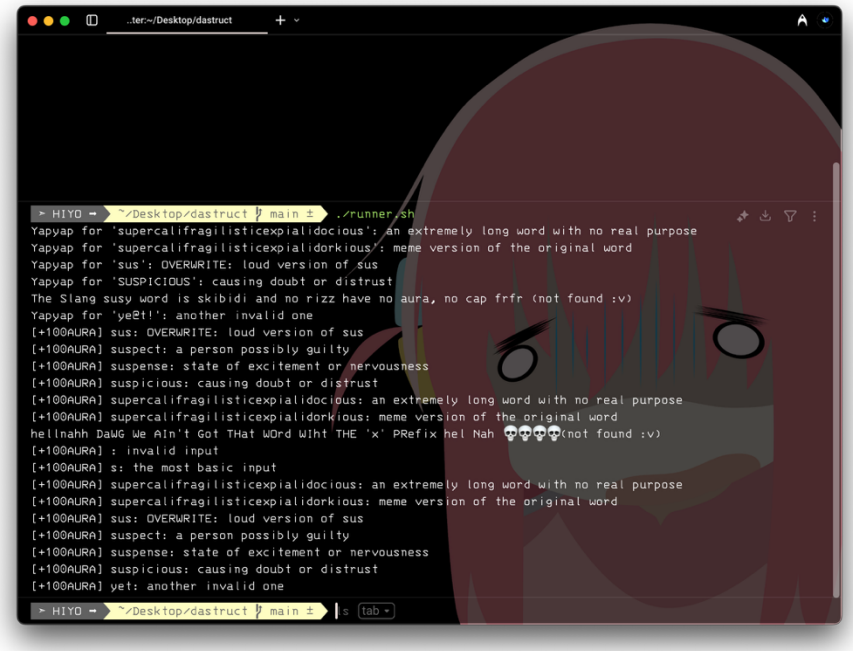
```
// Insertion
  insert(root, "supercalifragilisticexpialidocious",
        "an extremely long word with no real purpose");
  insert(root, "supercalifragilisticexpialidorkious",
        "meme version of the original word");
  insert(root, "sus", "something suspicious");
  insert(root, "suspense", "state of excitement or nervousness");
  insert(root, "suspect", "a person possibly guilty");
  insert(root, "suspicious", "causing doubt or distrust");
  insert(root, "SUS", "OVERWRITE: loud version of sus");
  insert(root, "s", "the most basic input");

  // Invalid word (should be ignored silently
or skipped)
  insert(root, "123", "invalid input");
  insert(root, "ye@t!", "another invalid
one");

  // Searches
  search(root,
        "supercalifragilisticexpialidocious"); //
✅ should return definition
  search(
     root,
     "supercalifragilisticexpialidorkious"); //
✅ should return meme version
  search(root, "sus");        // ✅ should show
"OVERWRITE: loud version of sus"
  search(root, "SUSPICIOUS"); // ✅ should match (case-insensitive)
  search(root, "susy");       // ❌ should not be found
  search(root, "ye@t!");      // ❌ invalid characters ignored, no result

  // Prefix tests
  searchByPrefix(root,
              "sus"); // should list: sus, suspense, suspect, suspicious
  searchByPrefix(root, "superc"); // should list the two long words
  searchByPrefix(root, "x");      // ❌ no match

  // Sorted dump
  printAllWords(root, buffer, 0);
  // Insert random popular slang
```



```
    insert(root, "gg", "good game");
    insert(root, "ggwp", "good game, well played");
    insert(root, "glhf", "good luck, have fun");
    insert(root, "ez", "too easy");
    insert(root, "rekt", "completely destroyed");
    insert(root, "cringe", "something embarrassing or awkward");
    insert(root, "based", "unapologetically true or cool");
    insert(root, "mid", "something mediocre or average");
    insert(root, "goat", "greatest of all time");
    insert(root, "cap", "lie or exaggeration");
    insert(root, "nocap", "for real, no lie");

    // Search tests
    search(root, "gg");   // → good game
    search(root, "glhf"); // → good luck, have fun
    search(root, "mid");  // → something mediocre or average
    search(root, "GOAT"); // → case-insensitive match

    // Prefix tests
    searchByPrefix(root, "g");  // → gg, ggwp, glhf, goat
    searchByPrefix(root, "gg"); // → gg, ggwp
    searchByPrefix(root, "z");  // → no match

    // Sorted
    printAllWords(root, buffer, 0);
```