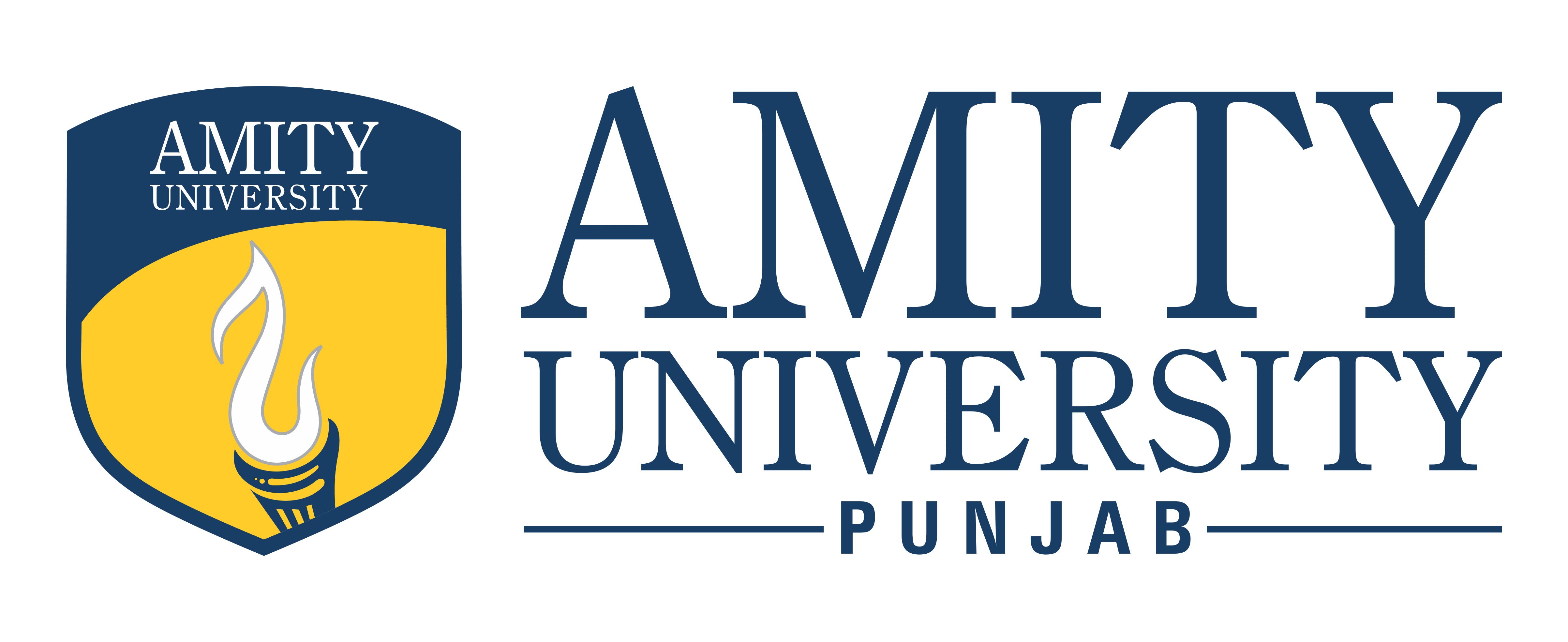
**LAB MANUAL**

**COMPILER DESIGN**

**CSE309**



**SESSION: 2022-2026**

**B. TECH-CSE-B**

**SEMESTER-6th**

**SUBMITTED TO: SUBMITTED BY:**

**Dr. Law Kumar Singh Deepali**

**ASET, Amity University A25305222126**

**Mohali**

1. **Write a program to make a calculator in C**

#include <stdio.h>

int main() {

    char operator;

    double num1, num2, result;

    // Display available operations

    printf("Simple Calculator in C\n");

    printf("Choose an operation (+, -, \*, /): ");

    scanf(" %c", &operator);

    // Get user input for numbers

    printf("Enter two numbers: ");

    scanf("%lf %lf", &num1, &num2);

    // Perform operation based on user choice

    switch (operator) {

        case '+':

            result = num1 + num2;

            break;

        case '-':

            result = num1 - num2;

            break;

        case '\*':

            result = num1 \* num2;

            break;

        case '/':

            if (num2 != 0)

                result = num1 / num2;

            else {

                printf("Error! Division by zero is not allowed.\n");

                return 1;

            }

            break;

        default:

            printf("Invalid operator! Please use +, -, \*, or /.\n");

            return 1;

    }

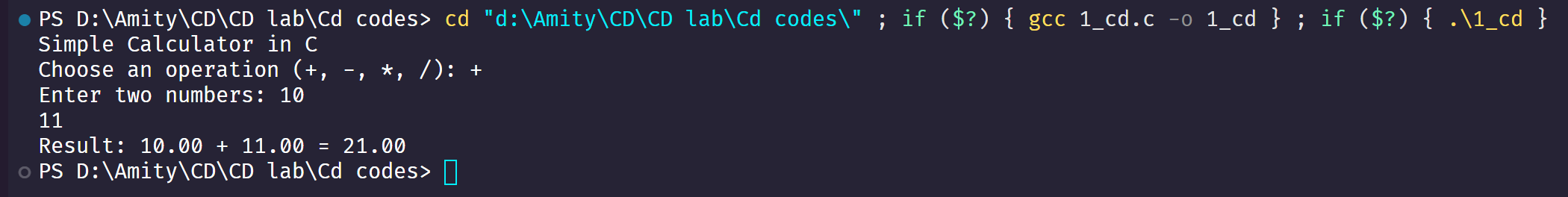
    // Display the result

    printf("Result: %.2lf %c %.2lf = %.2lf\n", num1, operator, num2, result);

    return 0;

}

**Output**

****

1. **Write a program for matrix addition in C**

#include <stdio.h>

#define SIZE 3

int main() {

    int matrix1[SIZE][SIZE], matrix2[SIZE][SIZE], result[SIZE][SIZE];

    int i, j;

    // Input first matrix

    printf("Enter elements of first %dx%d matrix:\n", SIZE, SIZE);

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            scanf("%d", &matrix1[i][j]);

        }

    }

    // Input second matrix

    printf("Enter elements of second %dx%d matrix:\n", SIZE, SIZE);

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            scanf("%d", &matrix2[i][j]);

        }

    }

    // Perform matrix addition

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            result[i][j] = matrix1[i][j] + matrix2[i][j];

        }

    }

    // Print the resulting matrix

    printf("Resultant Matrix after Addition:\n");

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            printf("%d ", result[i][j]);

        }

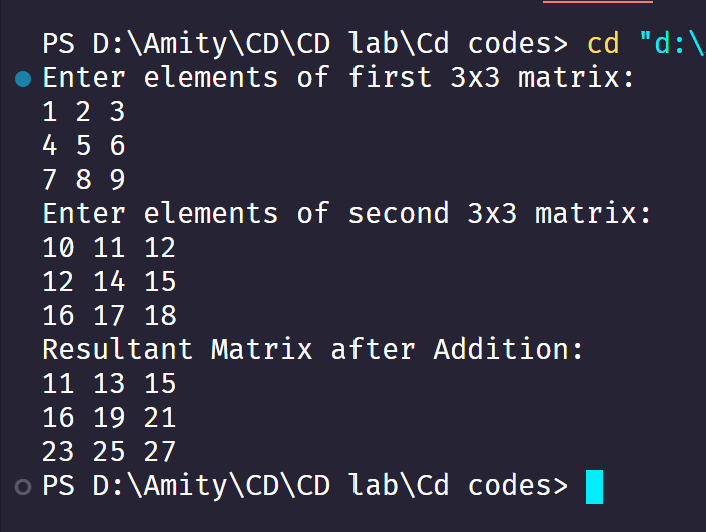
        printf("\n");

    }

    return 0;

}

**Output**

****

1. **Write a program for matrix multiplication in C**

#include <stdio.h>

#define SIZE 3 *// Define the size of the matrices*

int main() {

    int matrix1[SIZE][SIZE], matrix2[SIZE][SIZE], result[SIZE][SIZE];

    int i, j, k;

*// Input first matrix*

    printf("Enter elements of first *%d*x*%d* matrix:\n", SIZE, SIZE);

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            scanf("*%d*", &matrix1[i][j]);

        }

    }

*// Input second matrix*

    printf("Enter elements of second *%d*x*%d* matrix:\n", SIZE, SIZE);

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            scanf("*%d*", &matrix2[i][j]);

        }

    }

*// Initialize result matrix to 0*

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            result[i][j] = 0;

        }

    }

*// Perform matrix multiplication*

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            for (k = 0; k < SIZE; k++) {

                result[i][j] += matrix1[i][k] \* matrix2[k][j];

            }

        }

    }

*// Print the resultant matrix*

    printf("Resultant Matrix after Multiplication:\n");

    for (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            printf("*%d* ", result[i][j]);

        }

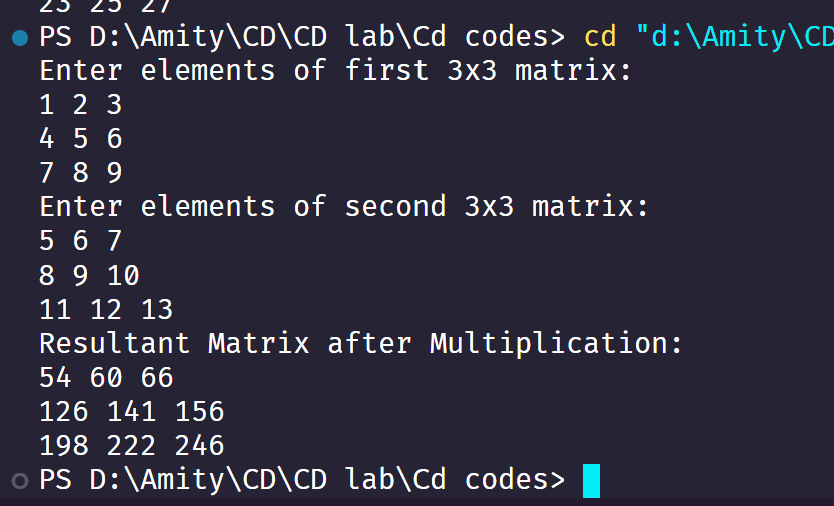
        printf("\n");

    }

    return 0;

}

**Output**

****

1. **Write a program to check if a string is constant or not**

#include <stdio.h>

#include <string.h>

int is\_string\_constant(const char \*str) {

    return 1;

}

int main() {

    // String literal (constant, stored in read-only memory)

    const char \*str1 = "Hello, World!";

    // Modifiable string (not constant)

    char str2[] = "Hello, World!";

    printf("Checking string constants:\n");

    // str1 is a pointer to a string literal (should be treated as constant)

    printf("str1 (%s): %s\n", str1,

           is\_string\_constant(str1) ? "Constant" : "Not Constant");

    // str2 is a modifiable array (not constant)

    printf("str2 (%s): %s\n", str2,

           is\_string\_constant(str2) ? "Constant" : "Not Constant");

    // Important note:

    printf("\nNote: In C, string literals are constants by convention,\n");

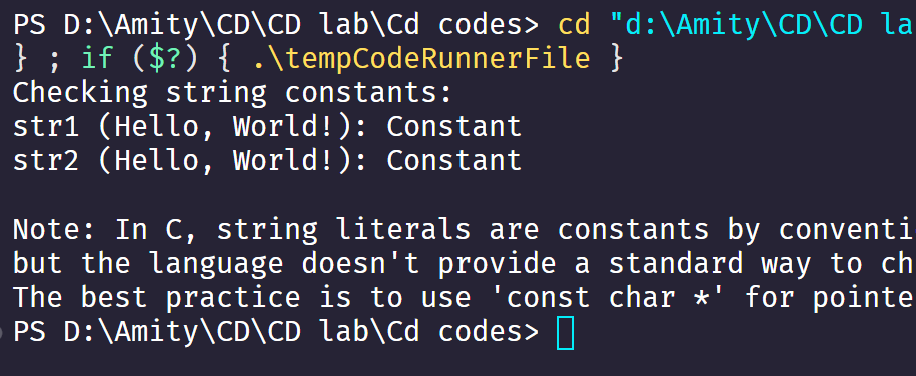
    printf("but the language doesn't provide a standard way to check this at runtime.\n");

    printf("The best practice is to use 'const char \*' for pointers to string literals.\n");

    return 0;

}

**Output**

****

1. **Write a program to count spaces and number of lines in C**

#include <stdio.h>

int main() {

    char ch;

    int spaces = 0, lines = 0;

    printf("Enter text (Press Ctrl+D to end input in Linux/Mac, Ctrl+Z in Windows):\n");

    while ((ch = getchar()) != EOF) {

        if (ch == ' ')

            spaces++;

        else if (ch == '\n')

            lines++;

    }

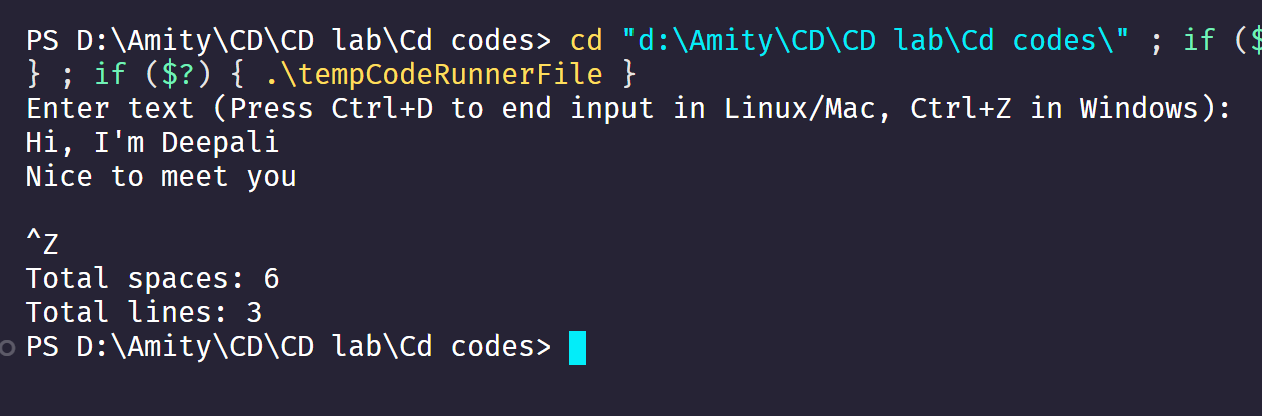
    printf("Total spaces: %d\n", spaces);

    printf("Total lines: %d\n", lines);

    return 0;

}

**Output**

****

1. **Write a program to check identifiers in C**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdbool.h>

// List of C keywords (partial list)

const char \*keywords[] = {

    "auto", "break", "case", "char", "const", "continue", "default", "do",

    "double", "else", "enum", "extern", "float", "for", "goto", "if",

    "int", "long", "register", "return", "short", "signed", "sizeof", "static",

    "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while"

};

bool is\_keyword(const char \*str) {

    int num\_keywords = sizeof(keywords) / sizeof(keywords[0]);

    for (int i = 0; i < num\_keywords; i++) {

        if (strcmp(str, keywords[i]) == 0) {

            return true;

        }

    }

    return false;

}

bool is\_valid\_identifier(const char \*str) {

    // Check if empty string

    if (str == NULL || \*str == '\0') {

        return false;

    }

    // Check first character

    if (!isalpha(str[0]) && str[0] != '\_') {

        return false;

    }

    // Check remaining characters

    for (int i = 1; str[i] != '\0'; i++) {

        if (!isalnum(str[i]) && str[i] != '\_') {

            return false;

        }

    }

    // Check if it's a keyword

    if (is\_keyword(str)) {

        return false;

    }

    return true;

}

int main() {

    char input[100];

    printf("Enter an identifier to check: ");

    scanf("%99s", input);

    if (is\_valid\_identifier(input)) {

        printf("'%s' is a valid C identifier.\n", input);

    } else {

        printf("'%s' is NOT a valid C identifier.\n", input);

    }

    // Test cases

    printf("\nTest cases:\n");

    const char \*test\_cases[] = {"valid", "\_valid", "123invalid", "invalid-name",

                               "int", "Valid123", "\_123\_valid\_", "space not valid"};

    int num\_cases = sizeof(test\_cases) / sizeof(test\_cases[0]);

    for (int i = 0; i < num\_cases; i++) {

        printf("%-15s : %s\n", test\_cases[i],

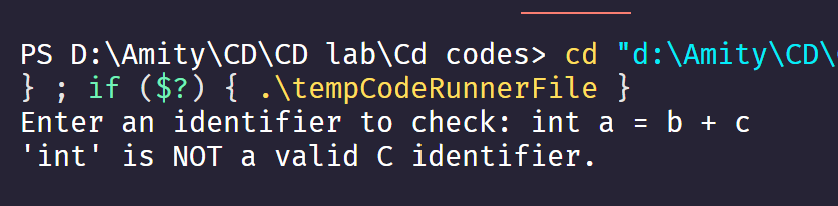
               is\_valid\_identifier(test\_cases[i]) ? "Valid" : "Invalid");

    }

    return 0;

}

**Output**

****

1. **Write a program to check keywords in C**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

// List of C keywords

const char \*keywords[] = {

    "auto", "break", "case", "char", "const", "continue", "default", "do", "double", "else",

    "enum", "extern", "float", "for", "goto", "if", "inline", "int", "long", "register",

    "restrict", "return", "short", "signed", "sizeof", "static", "struct", "switch", "typedef",

    "union", "unsigned", "void", "volatile", "while", "\_Alignas", "\_Alignof", "\_Atomic", "\_Bool",

    "\_Complex", "\_Generic", "\_Imaginary", "\_Noreturn", "\_Static\_assert", "\_Thread\_local"

};

#define KEYWORDS\_COUNT (sizeof(keywords) / sizeof(keywords[0]))

// Function to check if a word is a C keyword

bool isKeyword(const char \*word) {

    for (int i = 0; i < KEYWORDS\_COUNT; i++) {

        if (strcmp(word, keywords[i]) == 0) {

            return true;

        }

    }

    return false;

}

int main() {

    char word[50];

    printf("Enter a word: ");

    scanf("%s", word);

    if (isKeyword(word)) {

        printf("'%s' is a C keyword.\n", word);

    } else {

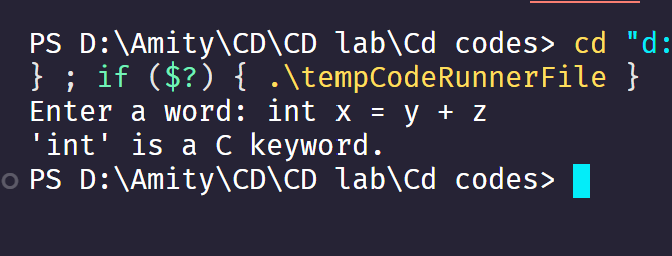
        printf("'%s' is NOT a C keyword.\n", word);

    }

    return 0;

}

**Output**

****

1. **Write a menu based program to check keyword, identifier, space and constant in C**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdbool.h>

// List of C keywords

const char \*keywords[] = {

    "auto", "break", "case", "char", "const", "continue", "default", "do",

    "double", "else", "enum", "extern", "float", "for", "goto", "if",

    "int", "long", "register", "return", "short", "signed", "sizeof", "static",

    "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while"

};

// Function prototypes

bool is\_keyword(const char \*str);

bool is\_valid\_identifier(const char \*str);

bool is\_whitespace(char ch);

bool is\_constant(const char \*str);

void print\_menu();

int main() {

    char input[100];

    int choice;

    do {

        print\_menu();

        printf("Enter your choice (1-5): ");

        scanf("%d", &choice);

        getchar(); // Consume newline

        switch(choice) {

            case 1: // Check keyword

                printf("Enter a word to check if it's a C keyword: ");

                fgets(input, sizeof(input), stdin);

                input[strcspn(input, "\n")] = '\0'; // Remove newline

                if(is\_keyword(input)) {

                    printf("'%s' is a C keyword.\n", input);

                } else {

                    printf("'%s' is NOT a C keyword.\n", input);

                }

                break;

            case 2: // Check identifier

                printf("Enter a string to check if it's a valid C identifier: ");

                fgets(input, sizeof(input), stdin);

                input[strcspn(input, "\n")] = '\0';

                if(is\_valid\_identifier(input)) {

                    printf("'%s' is a valid C identifier.\n", input);

                } else {

                    printf("'%s' is NOT a valid C identifier.\n", input);

                }

                break;

            case 3: // Check whitespace

                printf("Enter a character to check for whitespace: ");

                char ch = getchar();

                if(is\_whitespace(ch)) {

                    printf("'%c' is a whitespace character.\n", ch);

                } else {

                    printf("'%c' is NOT a whitespace character.\n", ch);

                }

                break;

            case 4: // Check constant

                printf("Enter a string to check if it's a constant: ");

                fgets(input, sizeof(input), stdin);

                input[strcspn(input, "\n")] = '\0';

                if(is\_constant(input)) {

                    printf("'%s' is a constant.\n", input);

                } else {

                    printf("'%s' is NOT a constant.\n", input);

                }

                break;

            case 5: // Exit

                printf("Exiting program...\n");

                break;

            default:

                printf("Invalid choice! Please try again.\n");

        }

        if(choice != 5) {

            printf("\nPress Enter to continue...");

            getchar(); // Wait for user

        }

    } while(choice != 5);

    return 0;

}

// Function to check if a string is a C keyword

bool is\_keyword(const char \*str) {

    int num\_keywords = sizeof(keywords) / sizeof(keywords[0]);

    for(int i = 0; i < num\_keywords; i++) {

        if(strcmp(str, keywords[i]) == 0) {

            return true;

        }

    }

    return false;

}

// Function to check if a string is a valid C identifier

bool is\_valid\_identifier(const char \*str) {

    if(str == NULL || \*str == '\0') {

        return false;

    }

    if(!isalpha(str[0]) && str[0] != '\_') {

        return false;

    }

    for(int i = 1; str[i] != '\0'; i++) {

        if(!isalnum(str[i]) && str[i] != '\_') {

            return false;

        }

    }

    if(is\_keyword(str)) {

        return false;

    }

    return true;

}

// Function to check if a character is whitespace

bool is\_whitespace(char ch) {

    return ch == ' ' || ch == '\t' || ch == '\n' || ch == '\r' || ch == '\f' || ch == '\v';

}

// Function to check if a string is a constant

bool is\_constant(const char \*str) {

    if(str == NULL || \*str == '\0') {

        return false;

    }

    // Check for integer constant

    bool is\_numeric = true;

    for(int i = 0; str[i] != '\0'; i++) {

        if(!isdigit(str[i])) {

            is\_numeric = false;

            break;

        }

    }

    if(is\_numeric) return true;

    // Check for string literal (in double quotes)

    if(str[0] == '"' && str[strlen(str)-1] == '"') {

        return true;

    }

    // Check for character constant (in single quotes)

    if(str[0] == '\'' && str[strlen(str)-1] == '\'' && strlen(str) == 3) {

        return true;

    }

    return false;

}

// Function to display menu

void print\_menu() {

    printf("\n===== C Code Element Analyzer =====\n");

    printf("1. Check if a word is a C keyword\n");

    printf("2. Check if a string is a valid C identifier\n");

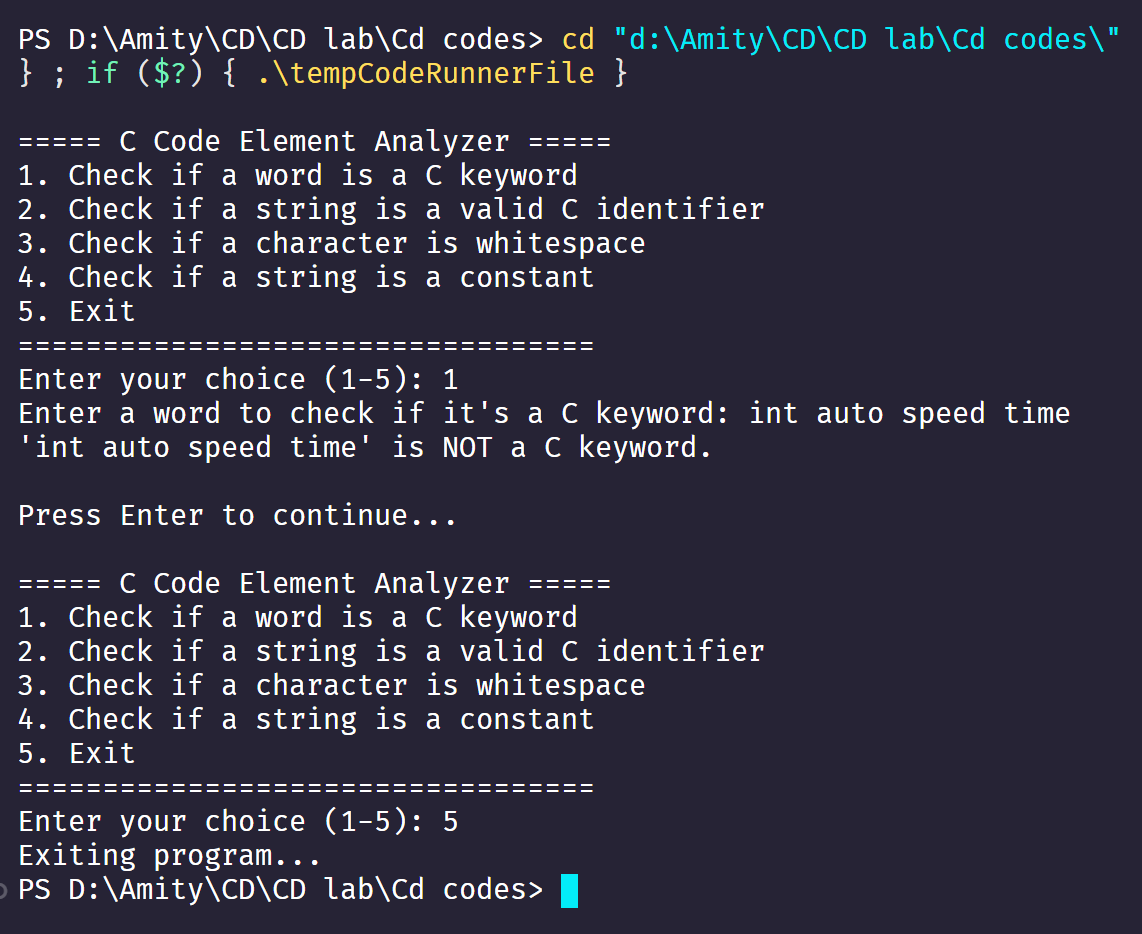
    printf("3. Check if a character is whitespace\n");

    printf("4. Check if a string is a constant\n");

    printf("5. Exit\n");

    printf("==================================\n");

**Output**

****

1. **Write a program for left recursion in C**

#include <stdio.h>

*// Left-recursive function example*

void leftRecursive(int *n*) {

    if (*n* <= 0) {

        return;

    }

    printf("*%d* ", *n*);

    leftRecursive(*n* - 1); *// Left recursion*

}

int main() {

    int num;

    printf("Enter a number: ");

    scanf("*%d*", &num);

    printf("Left-recursive call output: ");

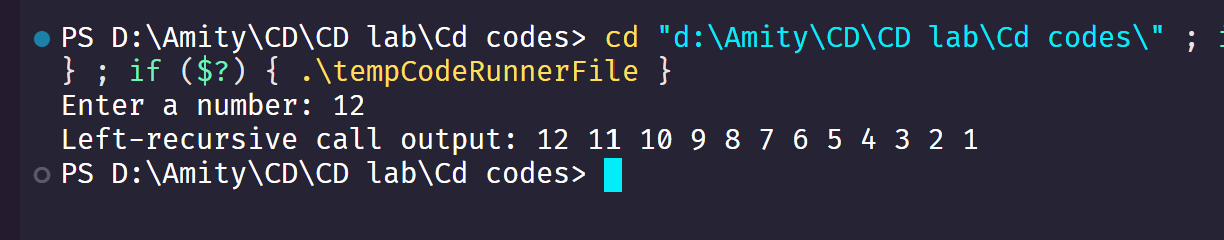
    leftRecursive(num);

    printf("\n");

    return 0;

}

**Output**

****