1A

Simple Calculator in C

#include <stdio.h>

int main() { char op; double num1, num2, result;

```
printf("Enter an operator (+, -, *, /): ");
scanf(" %c", &op);

printf("Enter two numbers: ");
scanf("%lf %lf", &num1, &num2);

switch (op) {
    case '+':
        result = num1 + num2;
        break;
    case '-':
        result = num1 - num2;
        break;
    case '*':
        result = num1 * num2;
        break;
    case '/':
        result = (num2 != 0) ? num1 / num2 : 0;
        break;
    default:
        printf("Invalid operator\n");
        return 1;
}

printf("Result: %.2lf\n", result);
return 0;
```

}

1B

1. Lex code to count total number of characters

```
%{
int char_count = 0;
%}
%%
. { char_count++; } // Increment count for every character
\n { char_count++; } // Count newline characters too
%%
```

```
int yywrap() { return 1; }
int main() {
    yylex();
    printf("Total number of characters: %d\n", char_count);
    return 0;
}
```

2. Lex Program to count number of words

```
%{
int word_count = 0;
%}
%%
[^\t\n ]+ { word_count++; } // Match any non-space character sequence as a word
\n { } // Ignore newlines
%%
int yywrap() { return 1; }
int main() {
    yylex();
    printf("Total number of words: %d\n", word_count);
    return 0;
}
```

3. A simple lexer that recognizes identifiers, numbers, and operators

```
%{
#include <stdio.h>
%}
%%
[a-zA-Z_][a-zA-Z0-9_]* { printf("IDENTIFIER: %s\n", yytext); }
[0-9]+ { printf("NUMBER: %s\n", yytext); }
[+\-*/=] { printf("OPERATOR: %s\n", yytext); }
. { printf("OTHER: %s\n", yytext); }
%%
int main() {
    yylex();
    return 0;
}
int yywrap() { return 1; }
```

4. Lex program to count the number of lines, spaces and tabs

```
%{
int line_count = 0, space_count = 0, tab_count = 0;
%}
%%
\n { line_count++; } // Count new lines
```

```
" " { space_count++; } // Count spaces
"\t" { tab_count++; } // Count tabs
%%
int yywrap() { return 1; }
int main() {
    yylex();
    printf("Total lines: %d\n", line_count);
    printf("Total spaces: %d\n", space_count);
    printf("Total tabs: %d\n", tab_count);
    return 0;
}
```

5. Lex program to count the frequency of the given word in a file

```
%{
#include <stdio.h>
#include <string.h>

int count = 0;
char *word;
%}

%%
[a-zA-Z]+ {
    if(strcmp(yytext, word) == 0)
        count++;
}
. { }
\n { }
%%

int yywrap() { return 1; }

int main(int argc, char *argv[]) {
    if(argc != 2) {
        printf("Usage: %s word\n", argv[0]);
        return 1;
    }

    word = argv[1];
    yylex();
    printf("Frequency of the word '%s': %d\n", word, count);
    return 0;
}
```

6. LEX program to add line numbers to a given file

```
%{
int line_num = 1;
%}

%%
^.* { printf("%4d\t%s", line_num++, yytext); }
\n { printf("\n"); }
%%

int yywrap() { return 1; }

int main(int argc, char *argv[]) {
    if(argc > 1) {
        FILE *file = fopen(argv[1], "r");
        if(!file) {
            printf("Could not open file %s\n", argv[1]);
            return 1;
        }
        yyin = file;
    }

    yylex();
    return 0;
}
```

To compile and run these programs: 1. Save the code in a file with a `.l` extension (e.g., `char_count.l`) 2. Compile it using: `flex filename.l    gcc lex.yy.c -o program_name -lfl` 3. Run the program: `./program_name [input_file]  # For programs that accept a file    ./program_name # For programs that read from stdin`

For program #5, you would run it like:

```
./word_frequency target_word < input_file
```

For program #6, you would run it like:

```
./line_number input_file
```