Set-8

1. All languages have Grammar. When people frame a sentence we usually say whether the sentence is framed as per the rules of the Grammar or Not. Similarly use the same ideology , implement to check whether the given input string is satisfying the grammar or not .

Code:

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

char input[100];
int pos = 0;

// Function prototypes
void S();
void match(char expected);

// Match and consume expected character
void match(char expected) {
  if (input[pos] == expected) {
    pos++;
  } else {
    printf("Error: Unexpected character '%c'\n", input[pos]);
    exit(1);
  }
}

// Grammar rule implementation (Example Grammar: S -> aSb | ab)
void S() {
  if (input[pos] == 'a') {
    pos++;
    S();
    if (input[pos] == 'b') {
      pos++;
    } else {
      printf("Error: Expected 'b' at position %d\n", pos);
      exit(1);
    }
  }
```

```
}

int main() {
  printf("Enter a string: ");
  scanf("%s", input);

  S();

  if (input[pos] == '\0') {
    printf("String satisfies the grammar!\n");
  } else {
    printf("Error: Unparsed input remaining\n");
  }

  return 0;
}
```

2. Write a C program for implementing a Lexical Analyzer to Count the number of characters, words, and lines.

Code:

```
#include <stdio.h>
#include <ctype.h>

int main() {
  FILE *file;
  char filename[100], ch;
  int characters = 0, words = 0, lines = 0;
  int inWord = 0;

  // Get the filename from user
  printf("Enter the filename: ");
  scanf("%s", filename);

  // Open file
  file = fopen(filename, "r");
  if (file == NULL) {
    printf("Cannot open file %s\n", filename);
    return 1;
  }

  // Read file character by character
  while ((ch = fgetc(file)) != EOF) {
```

```
        characters++;

        if (ch == '\n') {
            lines++;
        }

        if (isspace(ch)) {
            inWord = 0;
        } else if (!inWord) {
            inWord = 1;
            words++;
        }
    }

    // Close file
    fclose(file);

    // Print results
    printf("Characters: %d\n", characters);
    printf("Words: %d\n", words);
    printf("Lines: %d\n", lines);

    return 0;
}
```

3. Write a LEX program to recognise numbers and words in a statement. Pooja is a small girl of age 3 always fond of games. Due to the pandemic, she was not allowed to play outside. So her mother designs a gaming event by showing a flash card. Pooja has to separate the numbers in one list and words in another list shown in the flash card.

   Code:

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

%%
// Match words
[a-zA-Z]+ { printf("Word: %s\n", yytext); }

// Match numbers
[0-9]+ { printf("Number: %s\n", yytext); }
```

```
// Ignore spaces and new lines
[ \t\n]+   ;

%%
int main() {
   printf("Enter a statement:\n");
   yylex();
   return 0;
}
```

4. Write a LEX program to identify and count positive and negative numbers.

Code:

```
%{
#include <stdio.h>
int positive_count = 0, negative_count = 0;
%}

%%
// Match positive numbers
[+]?[0-9]+ { positive_count++; printf("Positive Number: %s\n", yytext); }

// Match negative numbers
-[0-9]+ { negative_count++; printf("Negative Number: %s\n", yytext); }

// Ignore spaces and new lines
[ \t\n]+   ;

%%
int main() {
   printf("Enter numbers: \n");
   yylex();
   printf("Total Positive Numbers: %d\n", positive_count);
```

```
    printf("Total Negative Numbers: %d\n", negative_count);
    return 0;
}
```