Practical 5

Title: (a) WALEx Program to count words, characters, lines, Vowels and consonants from given input

Hint: This program will take a string input and then count the number of words, characters, lines, vowels, and consonants in that input.

Program:

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

int characters = 0, words = 1, lines = 1, vowels = 0, consonants = 0;

int isVowel(char ch) {
   ch = tolower(ch);
   return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u');
}
%}

%%

[a-zA-Z] {
   characters++;     // Count characters
   if (isVowel(yytext[0])) {
      vowels++;      // Count vowels
   } else {
      consonants++;  // Count consonants
   }
}

[ \t] {
   characters++;    // Count characters
}

\n {
   lines++;         // Count lines
   characters++;    // Count newline as a character
}

[.] {
   characters++;    // Period marks the end of input
   return 0;        // End lexing when a period is found
```

```
}

%%

int main() {
    printf("Enter text (end input with a period '.'): \n");
    yylex(); // Start scanning and tokenizing input

    printf("Jay Dalsaniya \n");
    printf("92100103336 \n");
    printf("\nStatistics:\n");
    printf("Characters: %d\n", characters);
    printf("Words: %d\n", words);
    printf("Lines: %d\n", lines);
    printf("Vowels: %d\n", vowels);
    printf("Consonants: %d\n", consonants);

    return 0;
}

int yywrap() {
    return 1;
}
```

Output:

```
F:\sem 7\CD\practical5a>lex p5a.l

F:\sem 7\CD\practical5a>gcc lex.yy.c

F:\sem 7\CD\practical5a>a.exe
Enter text (end input with a period '.'):
hello jay dalsaniya.
Jay Dalsaniya
92100103336

Statistics:
Characters: 20
Words: 1
Lines: 1
Vowels: 7
Consonants: 10
```

(b) WALEx Program to generate string which is ending with zeros.

Hint: This program will take a string input and append a specific number of zeros to it, based on a given condition.

Program:

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

int num_zeros = 0; // Variable to store the number of zeros to append
%}

%%

[a-zA-Z0-9]+ {
// This pattern matches any alphanumeric string
// Get the length of the string
int len = yyleng;

// Determine the number of zeros to append
num_zeros = (len % 3); // Example condition: append zeros based on the length modulo 3

printf("Jay Dalsaniya \n");
printf("92100103336 \n");

// Print the original string
printf("Original String: %s\n", yytext);

// Print the string followed by the zeros
printf("Modified String: %s", yytext);

// Variable declaration outside of the loop
int i;
for (i = 0; i < num_zeros; i++) {
printf("0");
}
printf("\n");
}

.|\n  {
// Any other character (including new lines) is ignored
}
```

```
%%

int main(int argc, char **argv) {
yylex();
return 0;
}

int yywrap() {
return 1;
}
```

Output:

```
F:\sem 7\CD\practical5b>lex p5b.l

F:\sem 7\CD\practical5b>gcc lex.yy.c

F:\sem 7\CD\practical5b>a.exe
hello
Jay Dalsaniya
92100103336
Original String: hello
Modified String: hello00
```

Practical 6

Title: (a) WALex Program to generate Histogram of words

Hint: This program will take a string input and generate a histogram based on the length of each word.

Program:

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

char words[1000][50];
int counts[1000], n = 0, i;
%}

%%

[a-zA-Z]+   {
   for (i = 0; i < n && strcmp(words[i], yytext); i++);
   if (i < n)
      counts[i]++;
   else {
      strcpy(words[n], yytext);
      counts[n++] = 1;
   }
}
.|\n        ;  // Ignore other characters

%%

int main() {
   printf("Jay Dalsaniya \n");
   printf("92100103336 \n");
   printf("Enter the Sentence:\n");
   yylex();
   for (i = 0; i < n; i++)
      printf("%s: %d\n", words[i], counts[i]);
   return 0;
}

int yywrap() {
   return 1;
}
```

Output:

```
F:\sem 7\CD\practical6a>lex p6a.l

F:\sem 7\CD\practical6a>gcc lex.yy.c

F:\sem 7\CD\practical6a>a.exe
Enter the Sentence:
hello jayu , jayu is a mu student , mu in rajkot
^Z
hello: 1
jayu: 2
is: 1
a: 1
mu: 2
student: 1
in: 1
rajkot: 1
```

(b) WALex Program to remove single or multi line comments from C program

Hint: To remove comments from a C program using Lex

Single-line comments (// ...): Use //[^'\n']* to ignore everything until the end of the line.
Multi-line comments (/* ... */): Use /* to start and */ to end, employing a custom function to handle the removal of these comments.

Program:

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

int sl = 0;  // Counter for single-line comments
int ml = 0;  // Counter for multi-line comments

%}

%%

"//"[^\n]*  { sl++; }  // Match single-line comments and increment counter
```

```
"/\*"([^*]|\*+[^/*])*\*+"/"  { ml++; }  // Match multi-line comments and increment counter

%%

int yywrap() {
   return 1;
}
int main() {
   yyin = fopen("f1.c", "r");
   yyout = fopen("f2.c", "w");

   if (!yyin) {
      perror("Failed to open input file");
      return 1;
   }
   if (!yyout) {
      perror("Failed to open output file");
      fclose(yyin);
      return 1;
   }
   yylex();
   fclose(yyin);
   fclose(yyout);
   printf("Jay Dalsaniya \n");
   printf("92100103336 \n");
   printf("\nNumber of single line comments = %d\n", sl);
   printf("\nNumber of multiline comments = %d\n", ml);

   return 0;
}
```

Output:

```
F:\sem 7\CD\practical6b>lex p6b.l

F:\sem 7\CD\practical6b>gcc lex.yy.c

F:\sem 7\CD\practical6b>a.exe
Jay Dalsaniya
92100103336

Number of single line comments = 3

Number of multiline comments = 3
```

F1 file output:

f1.c

F: > sem 7 > CD > practical6b > f1.c

```c
#include <stdio.h>

int main() {
    /*
     * This program is a basic example of how to use
     * comments in C. The alignment of asterisks (*)
     * makes it easier to read and maintain.
     */

    // This is the start of the program execution
    /*
     This is a B-Batch 7tc4 students lab.
     */
    // Print a simple greeting message to the console
    printf("Hello, World!\n");

    /*
     * The return value is 0, which signals that
     * the program executed without any errors.
     */

    // End of the main function, return 0 indicates success
    return 0;
}

```

F2 file output:

f2.c

F: > sem 7 > CD > practical6b > f2.c

```c
#include <stdio.h>

int main() {



    printf("Hello, World!\n");



    return 0;
}
```

Practical 7

Title: WALex Program to check weather given statement is compound or simple.

Hint: To check whether a given statement is compound or simple using Lex

Program:

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

int flag = 0; // Flag to determine if the sentence is compound
%}

%%

and|or|but|because|if|then|nevertheless  { flag = 1; } // Set flag for compound sentence
[.?!] ;                                  // Match end of sentence punctuation
\n  { return 0; }                        // Return 0 on newline (end of input)
[ \t]+ ;                                 // Ignore whitespace
. ;                                      // Match any other single character (ignore)

%%

int main() {
   printf("Jay Dalsaniya \n");
   printf("92100103336 \n");

   printf("Enter the sentence:\n");
   yylex(); // Invoke the lexer
   if (flag == 0)
         printf("\nThis is a simple sentence.\n");
   else
      printf("\nThis is a compound sentence.\n");

   return 0;
}

int yywrap() {
   return 1; // Indicate end of input
}
```

Output:

```
F:\sem 7\CD\practical7>lex p7.l

F:\sem 7\CD\practical7>gcc lex.yy.c

F:\sem 7\CD\practical7>a.exe
Jay Dalsaniya
92100103336
Enter the sentence:
I will go to the park because I need some fresh air.

This is a compound sentence.

F:\sem 7\CD\practical7>a.exe
Jay Dalsaniya
92100103336
Enter the sentence:
I went to the park.

This is a simple sentence.
```