Compiler Design Assignment No: 1

Name: Sarthak Sanjay Pithe

PRN: 12210166

Division: D

Roll No: 09

1. Write Specification of LEX/FLEX Program.

**Blueprint of a LEX program:**

%{

/\* C Declarations and Headers \*/

#include <stdio.h>

%}

/\* Definitions Section \*/

DIGIT [0-9]

LETTER [a-zA-Z]

%%

/\* Rules Section \*/

{LETTER}+ { printf("Found an identifier: %s\n", yytext); }

{DIGIT}+ { printf("Found a number: %s\n", yytext); }

"+" { printf("Found a plus sign\n"); }

"-" { printf("Found a minus sign\n"); }

. { printf("Unknown character: %s\n", yytext); }

%%

/\* User Code Section \*/

int main() {

yylex(); /\* Call to lexical analyzer \*/

return 0;

}

int yywrap() {

return 1;

}

**Explanation per section:**

1. **Declarations Section (%{ ... %})**:
   * This section includes C headers and declarations. It allows you to import libraries like <stdio.h> for input/output operations.
2. **Definitions Section**:
   * Here, you define patterns using regular expressions. For example, DIGIT is a pattern that matches any digit ([0-9]), and LETTER matches any letter ([a-zA-Z]).
3. **Rules Section (%%)**:
   * This section defines the core rules for token recognition:
     + {LETTER}+: Matches one or more letters, which is an identifier.
     + {DIGIT}+: Matches one or more digits, recognizing a number.
     + "+", "-": Matches specific symbols like plus and minus.
     + .: Matches any other single character as an unknown symbol.
4. **User Code Section (%% after rules)**:
   * The main function starts the lexical analysis with a call to yylex().
   * yywrap() is a function that signals the end of input. It returns 1 to indicate completion.