CSA1447-COMPILER DESIGN FOR SYNTAX SMITH

PROGRAM 22-29

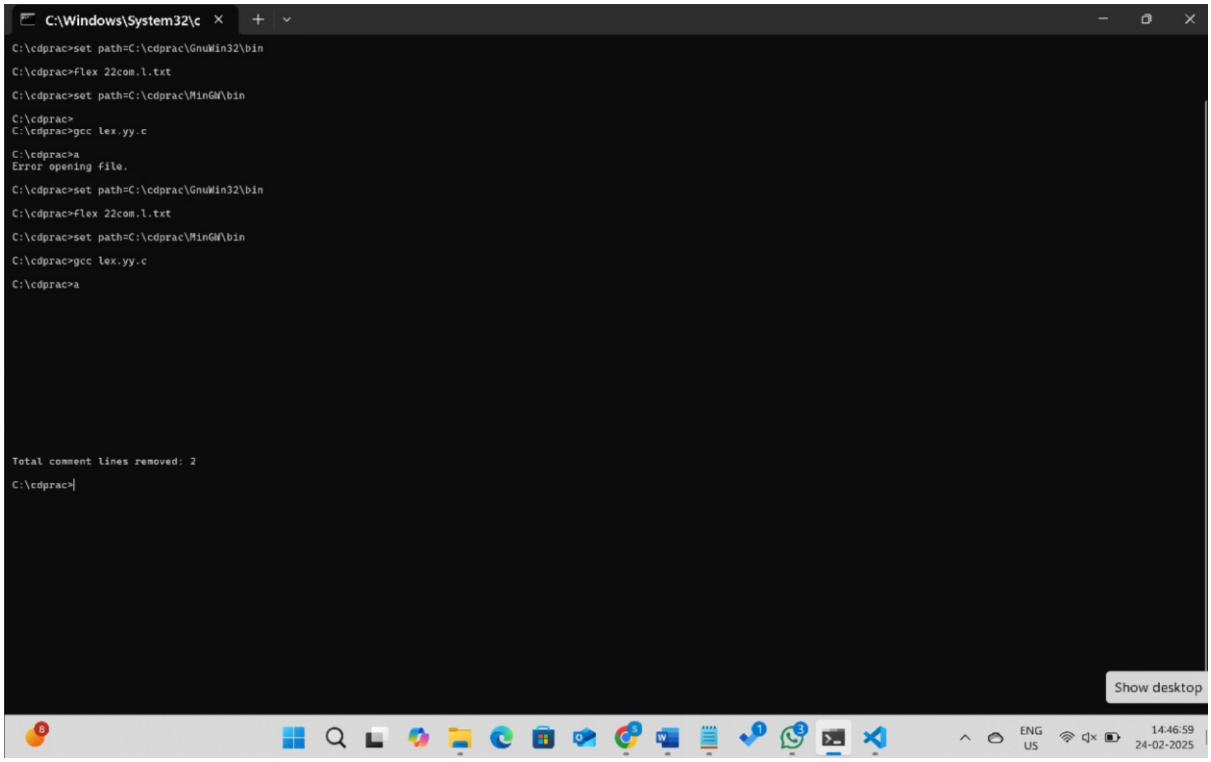
EXP-22

PROGRAM:

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

int comment_count = 0;
FILE *output;

%}

%%

"//".* { comment_count++; }  /* Single-line comment */
"/\*"(.|\n)*?"\*/" { comment_count++; }  /* Multi-line comment */

. { fputc(yytext[0], output); } /* Copy other content */

%%

int main() {
   FILE *input = fopen("input.c", "r");  // Open input file
   output = fopen("output.c", "w");  // Open output file to write modified content

   if (!input || !output) {
      printf("Error opening file.\n");
      return 1;
```

```
    }

    yyin = input;
    yylex();  // Process input file

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

    fclose(input);
    fclose(output);
    return 0;
}

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

OUTPUT:

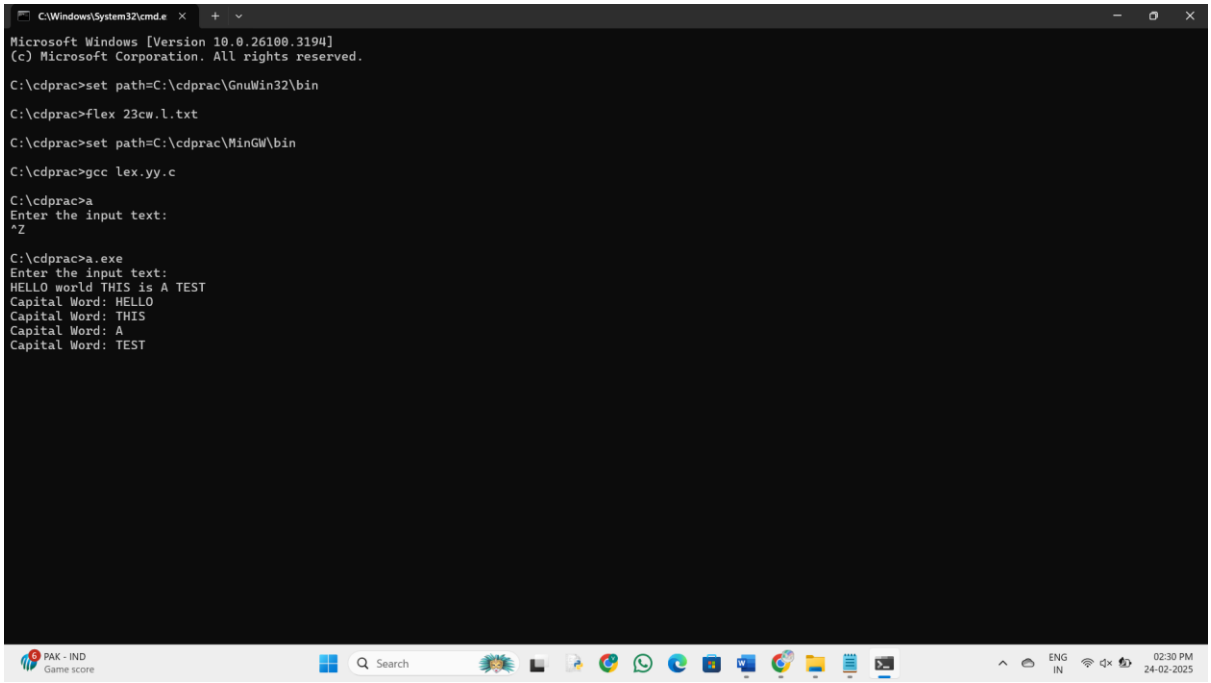


EXP-23

PROGRAM:

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

%%

[A-Z]+  { printf("Capital Word: %s\n", yytext); }  /* Matches capital words */
.|\n    { /* Ignore other characters */ }

%%

int main() {
    printf("Enter the input text:\n");
    yylex();
    return 0;
}

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

OUTPUT:



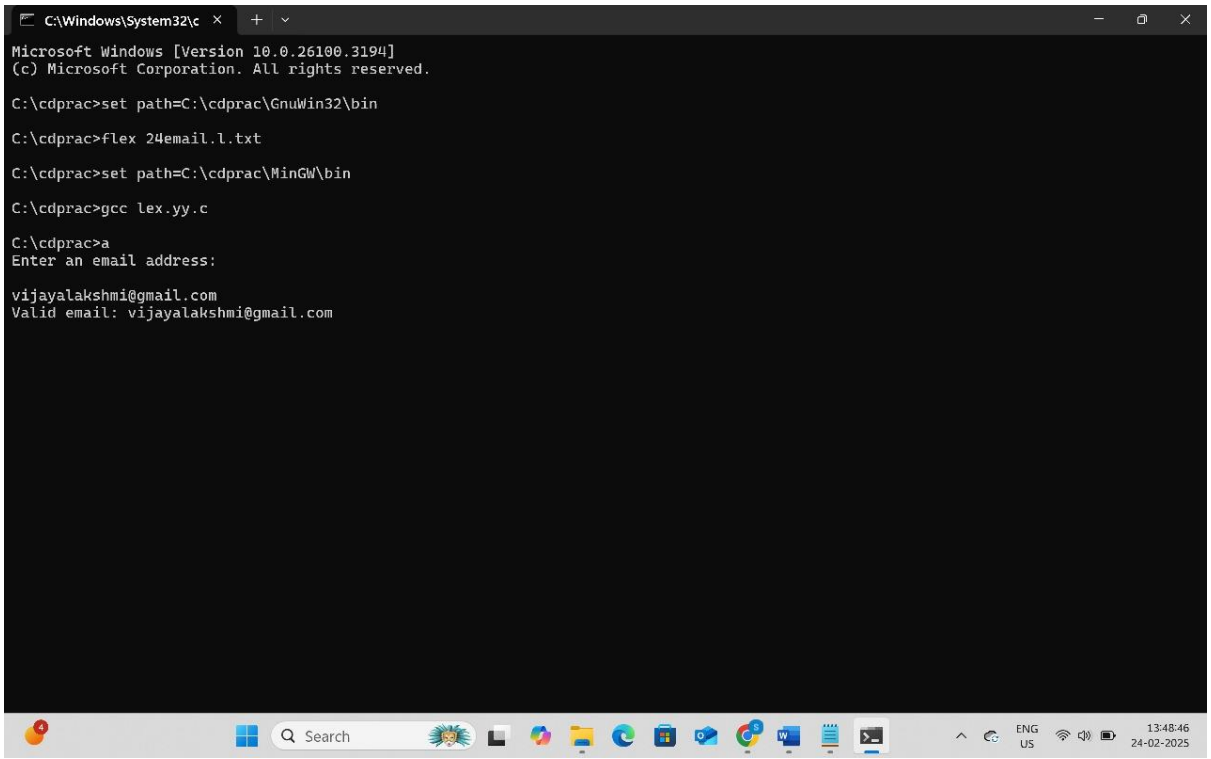
Exp-24

PROGRAM:

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

%%

[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}  { printf("Valid email: %s\n", yytext); }
[^ \t\n]+                                       { printf("Invalid email: %s\n", yytext); }
[ \t\n]                                         ;  /* Ignore whitespace */

%%

int main() {
```

```
    printf("Enter an email address: \n");
    yylex();
    return 0;
}

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

OUTPUT:



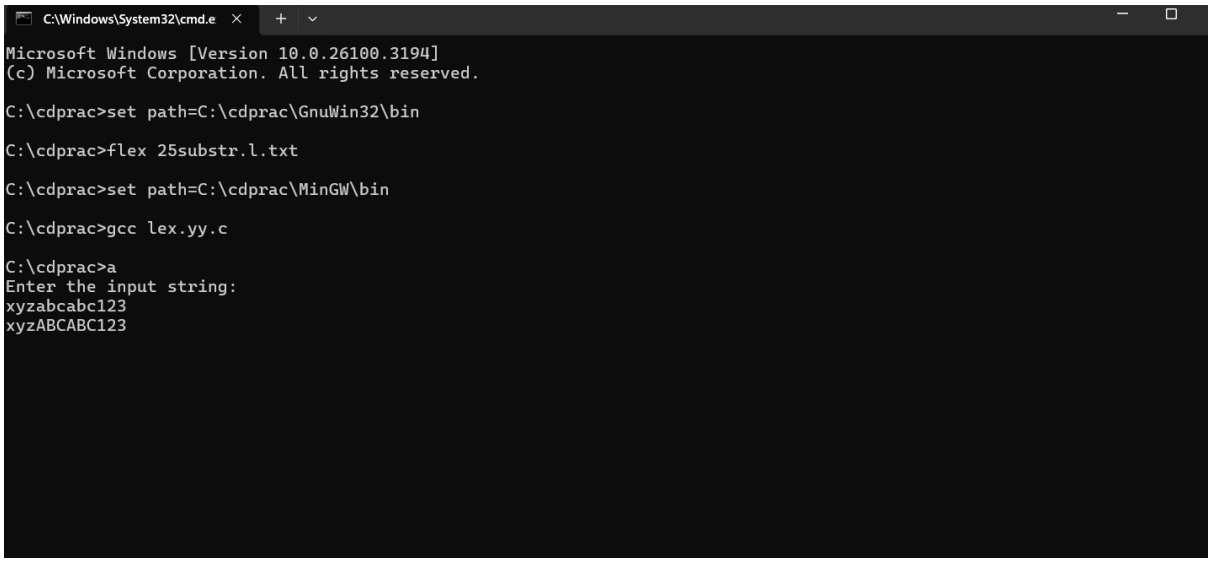
EXP-25

PROGRAM:

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

```
%%

abc     { printf("ABC"); }  /* Replace "abc" with "ABC" */
.|\n    { printf("%s", yytext); } /* Print other characters as they are */

%%

int main() {
    printf("Enter the input string:\n");
    yylex();
    return 0;
}

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

OUTPUT:



EXP-26

PROGRAM:

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

%%

[789][0-9]{9}  { printf("Valid Mobile Number: %s\n", yytext); }  /* Matches valid mobile numbers */
[0-9]+        { printf("Invalid Mobile Number: %s\n", yytext); }  /* Catches invalid numbers */
.|\n          { /* Ignore other characters */ }

%%

int main() {
   printf("Enter mobile numbers (separated by space or new lines):\n");
   yylex();
   return 0;
}

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

OUTPUT:

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\cdprac>set path=C:\cdprac\GnuWin32\bin

C:\cdprac>flex 26mv.l.txt

C:\cdprac>set path=C:\cdprac\MinGW\bin

C:\cdprac>gcc lex.yy.c

C:\cdprac>a
Enter mobile numbers (separated by space or new lines):
9876543210 1234567890 8123456789 7890456123 5678901234
Valid Mobile Number: 9876543210
Invalid Mobile Number: 1234567890
Valid Mobile Number: 8123456789
Valid Mobile Number: 7890456123
Invalid Mobile Number: 5678901234
```

EXP-27

PROGRAM:

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

%%

"#include" { printf("Preprocessor Directive: %s\n", yytext); } "int"|"void" { printf("Keyword: %s\n", yytext); } [a-zA-Z_][a-zA-Z0-9_]* { printf("Identifier: %s\n", yytext); } [0-9]+ { printf("Number: %s\n", yytext); } "=" { printf("Assignment Operator: %s\n", yytext); } "," { printf("Comma\n"); } ";" { printf("Semicolon\n"); } "(" | ")" { printf("Parenthesis: %s\n", yytext); } "{" | "}" { printf("Brace: %s\n", yytext); } .|\n { /* Ignore other characters */ }

%%

int main() { printf("Lexical Analysis Output:\n"); yylex(); return 0; }

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

Op:

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\cdprac>set path=C:\cdprac\GnuWin32\bin

C:\cdprac>flex 27lexa.l.txt

C:\cdprac>set path=C:\cdprac\MinGW\bin

C:\cdprac>gcc lex.yy.c

C:\cdprac>a
Lexical Analysis Output:
a
Identifier: a

=
Assignment Operator: =
*
b
Identifier: b
```

Exp-28:

%{ #include <stdio.h> int vowels = 0, consonants = 0; %}

%%

[AEIOUaeiou] { vowels++; } [a-zA-Z] { consonants++; } .|\n { /* Ignore other characters like spaces, digits, and punctuation */ }

%%

int main() { printf("Enter a sentence:\n"); yylex(); // Start lexical analysis printf("Number of vowels: %d\n", vowels); printf("Number of consonants: %d\n", consonants); return 0; }

int yywrap() { return 1; }

Op:

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\cdprac>set path=C:\cdprac\GnuWin32\bin

C:\cdprac>flex 28vow.l.txt

C:\cdprac>set path=C:\cdprac\MinGW\bin

C:\cdprac>gcc lex.yy.c

C:\cdprac>a
Enter a sentence:
hello this compiler design practical session
```

EXP-29

PROGRAM:

```
%{

#include <stdio.h>

#include <string.h>


// List of C keywords

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"

};


int is_keyword(char *word) {

    for (int i = 0; i < 32; i++) {

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

            return 1;
```

```
    }
    return 0;
}
%}

%%

[a-zA-Z_][a-zA-Z0-9_]* {
    if (is_keyword(yytext))
        printf("Keyword: %s\n", yytext);
    else
        printf("Identifier: %s\n", yytext);
}

[ \t\n]   ;  // Ignore spaces, tabs, and newlines
.         ;  // Ignore other characters

%%

int main() {
    printf("Enter the input code:\n");
    yylex();
    return 0;
}

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

OUTPUT:

```
C:\Windows\System32\c

Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\cdprac>set path=C:\cdprac\GnuWin32\bin

C:\cdprac>flex 29key.l.txt

C:\cdprac>set path=C:\cdprac\MinGW\bin

C:\cdprac>
C:\cdprac>gcc lex.yy.c

C:\cdprac>a
Enter the input code:
int num;
Keyword: int
Identifier: num
float value;
Keyword: float
Identifier: value
return 0;
Keyword: return
```