Lex Programming Practice Sheet Solutions

Lex Programming Practice Sheet Solutions

Level 1: Basics of Pattern Matching

1. Identifier Check
```lex
%{
#include <stdio.h>
%}
IDENTIFIER [a-zA-Z_][a-zA-Z0-9_]*
%%
{IDENTIFIER} { printf("Valid Identifier\n"); }
. { printf("Invalid Identifier\n"); }
%%
int main() {
    yylex();
    return 0;
}
```

2. Even or Odd Number
```lex
%%
[02468]$ { printf("Even Number\n"); }
[13579]$ { printf("Odd Number\n"); }
%%
```

3. Vowel or Consonant
```lex
%%
[aeiouAEIOU] { printf("Vowel\n"); }
[a-zA-Z] { printf("Consonant\n"); }
%%
```

4. Count Vowels, Consonants, Digits, and Special Characters
```lex
%{
int vowels=0, consonants=0, digits=0, specials=0;
%}
%%
[aeiouAEIOU] { vowels++; }
[a-zA-Z] { consonants++; }
[0-9] { digits++; }
. { specials++; }

<<EOF>> {
    printf("Vowels: %d\n", vowels);
```

```
    printf("Consonants: %d\n", consonants);
    printf("Digits: %d\n", digits);
    printf("Special Characters: %d\n", specials);
}
%%
```

5. Detect and Remove Whitespaces
```lex
%%
[ \t\n]+ ;
. { printf("%s", yytext); }
%%
```

Level 2: Validation Tasks

6. Email Validator
```lex
%%
[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4} { printf("Valid Email\n"); }
%%
```

7. Phone Number Validator
```lex
%%
[0-9]{10} { printf("Valid Phone Number\n"); }
%%
```

8. Date Validator (DD/MM/YYYY)
```lex
%%
(0[1-9]|[12][0-9]|3[01])\/(0[1-9]|1[0-2])\/([0-9]{4}) { printf("Valid Date\n"); }
%%
```

9. URL Validator
```lex
%%
(http|https):\/\/([a-zA-Z0-9.-]+)\/?.* { printf("Valid URL\n"); }
%%
```

10. IP Address Validator (IPv4)
```lex
%%
([0-9]{1,3}\.){3}[0-9]{1,3} { printf("Valid IP Address\n"); }
%%
```

Level 3: Tokenization and Analysis

11. Arithmetic Expression Analyzer
```lex
%%
[0-9]+ { printf("Number: %s\n", yytext); }
[+\-*/] { printf("Operator: %s\n", yytext); }
%%
```

12. Keyword Counter (C keywords)
```lex
%%
if|else|while|for { printf("Keyword: %s\n", yytext); }
%%
```

13. Lexical Analyzer for C Code
```lex
%%
[a-zA-Z_][a-zA-Z0-9_]* { printf("Identifier: %s\n", yytext); }
[0-9]+ { printf("Number: %s\n", yytext); }
%%
```

14. Detect Palindromes
```lex
%%
([a-zA-Z]+)\1 { printf("Palindrome: %s\n", yytext); }
%%
```

15. Longest Word Finder
```lex
%%
[a-zA-Z]+ { if(strlen(yytext) > max) { max = strlen(yytext); strcpy(longest, yytext); } }
<<EOF>> { printf("Longest Word: %s\n", longest); }
%%
```

Level 4: Output Formatting & Transformation

16. Add Line Numbers
```lex
%{
int line = 1;
%}
%%
.* { printf("%d: %s\n", line++, yytext); }
%%
```

17. Replace Multiple Spaces
```lex
%%
[ ]+ { printf(" "); }
```

```
%%
```

18. Convert to UPPERCASE
```lex
%%
[a-z] { printf("%c", toupper(yytext[0])); }
%%
```

19. Insert # at End of Lines
```lex
%%
.* { printf("%s#\n", yytext); }
%%
```

20. Remove Extra Occurrences
```lex
%%
([a-zA-Z]+) { if (!seen[yytext]) { seen[yytext] = 1; printf("%s ", yytext); } }
%%
```

Bonus Challenges

21. Check Balanced Parentheses
```lex
%%
[()] { balance += (yytext[0] == '(' ? 1 : -1); }
<<EOF>> { printf(balance == 0 ? "Balanced\n" : "Unbalanced\n"); }
%%
```

22. Mini wc Tool
```lex
%%
[a-zA-Z]+ { words++; }
. { chars++; }
\n { lines++; }
<<EOF>> { printf("Lines: %d, Words: %d, Chars: %d\n", lines, words, chars); }
%%
```

23. Detect HTML Tags
```lex
%%
<[^>]+> { printf("HTML Tag: %s\n", yytext); }
%%
```

24. Letter Frequency
```lex
%%
```

```
[a-zA-Z] { freq[tolower(yytext[0])]++; }
<<EOF>> { for (char c = 'a'; c <= 'z'; c++) printf("%c: %d\n", c, freq[c]); }
%%
```

25. Comment Remover
```lex
%%
//.*\n ;
/\*([^*]|\*+[^*/])*\*+/ ;
. { printf("%s", yytext); }
%%
```