LEX and YACC Code Collection

Q2. Count Lines, Spaces, Tabs, and Characters

```
%{
#include<stdio.h>
int lines = 0, spaces = 0, tabs = 0, chars = 0;
%}
%%
\n      { lines++; chars++; }
" "     { spaces++; chars++; }
\t      { tabs++; chars++; }
.       { chars++; }
%%
int yywrap() { return 1; }
int main(int argc, char* argv[]) {
    yylex();
    printf("Lines: %d\n", lines);
    printf("Spaces: %d\n", spaces);
    printf("Tabs: %d\n", tabs);
    printf("Other Characters: %d\n", chars - lines - spaces - tabs);
    printf("Total Characters: %d\n", chars);
    return 0;
}
```

Q3. Identify Valid C/C++ Identifier

```
%{
#include<stdio.h>
%}
%%
[a-zA-Z_][a-zA-Z0-9_]* { printf("%s is a valid identifier\n", yytext); }
.|\n                    {}
%%
int yywrap() { return 1; }
int main() {
    printf("Enter a string: ");
    yylex();
    return 0;
}
```

Q4. Identify Integer and Float Values

```
%{
#include <stdio.h>
%}
%%
[0-9]+          { printf("%s is an integer\n", yytext); }
[0-9]+\.[0-9]+  { printf("%s is a float\n", yytext); }
.|\n            {}
%%
int yywrap() { return 1; }
int main(void) {
    printf("Enter values to identify: \n");
    yylex();
    return 0;
}
```

Q5. Tokenize C-Fragment

```
%{
#include<stdio.h>
%}
%%
"int"|"else"|"if"   { printf("Keyword: %s\n", yytext); }
```

```
[a-zA-Z_][a-zA-Z0-9_]* { printf("Identifier: %s\n", yytext); }
"<"|">"|"=="|"!="|"=" { printf("Operator: %s\n", yytext); }
"+"|"-"|"*"|"/"     { printf("Operator: %s\n", yytext); }
[0-9]+              { printf("Constant: %s\n", yytext); }
";"|"("|")"|"{"|"}"   { printf("Separator: %s\n", yytext); }
[ \t\n]+            {}
.                   { printf("Unrecognized: %s\n", yytext); }
%%
int yywrap() { return 1; }
int main() {
    printf("Enter the C code fragment:\n");
    yylex();
    return 0;
}
```

Q6. Count Characters, Words, and Lines

```
%{
#include <stdio.h>
int char_count = 0;
int word_count = 0;
int line_count = 0;
%}
%%
\n        { char_count++; line_count++; }
[^\t\n ]+ { word_count++; char_count += yyleng; }
.         { char_count++; }
%%
int yywrap() { return 1; }
int main(int argc, char* argv[]) {
    FILE *fp = fopen("Input.txt", "r");
    if (!fp) {
        printf("Could not open Input.txt\n");
        return 1;
    }
    yyin = fp;
    yylex();
    printf("Characters: %d\n", char_count);
    printf("Words: %d\n", word_count);
    printf("Lines: %d\n", line_count);
    fclose(fp);
    return 0;
}
```

Q7. Replace Multiple White Spaces

```
%{
#include <stdio.h>
%}
%%
[ \t\n]+   { fprintf(yyout, " "); }
.          { ECHO; }
%%
int yywrap() { return 1; }
int main(int argc, char* argv[]) {
    yyin = fopen("Input.txt", "r");
    yyout = fopen("Output.txt", "w");
    yylex();
    fclose(yyin);
    fclose(yyout);
    return 0;
}
```

Q8. Remove Comments from a C Program

```
%{
#include <stdio.h>
%}
%%
"/*"([^*]|\*+[^*/])*\*+"/"   ;
```

```
"//"[^\n]* ;
.|\n                        ECHO;
%%
int yywrap(){ return 1; }
int main(int argc, char *argv[]){
    if (argc != 2){
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }
    FILE *f = fopen(argv[1], "r");
    if (!f){
        perror(argv[1]);
        return 1;
    }
    yyin = f;
    yylex();
    fclose(f);
    return 0;
}
```

Q9. Extract HTML Tags

```
%{
#include<stdio.h>
%}
%%
"<"[^>]*">" { fprintf(yyout, "%s\n", yytext); }
.|\n        ;
%%
int yywrap() { return 1; }
int main(int argc, char* argv[]) {
    if (argc < 3) {
        printf("Usage: ./a.out <input_html_file> <output_text_file>\n");
        return 1;
    }
    yyin = fopen(argv[1], "r");
    yyout = fopen(argv[2], "w");
    if (!yyin || !yyout) {
        printf("Error opening files.\n");
        return 1;
    }
    yylex();
    fclose(yyin);
    fclose(yyout);
    return 0;
}
```

Q10. DFA for Even 'a's and Even 'b's

```
%{
%}
%x B C D
%%
<INITIAL>a    BEGIN(B);
<INITIAL>b    BEGIN(C);
<B>a          BEGIN(A);
<B>b          BEGIN(D);
<C>a          BEGIN(D);
<C>b          BEGIN(A);
<D>a          BEGIN(C);
<D>b          BEGIN(B);
\n {
      if(YY_START==INITIAL) printf("Accepted (State A)\n");
      else printf("Rejected\n");
      BEGIN(INITIAL);
   }
. ;
%%
int main(){yylex();}
```

Q11. DFA for Third Last Element 'a'

```
%{
%}
%x A B C D E F G H
%%
<INITIAL>a    BEGIN(E);
<INITIAL>b    BEGIN(H);
<E>a          BEGIN(A);
<E>b          BEGIN(B);
<H>a          BEGIN(G);
<H>b          BEGIN(H);
<A>a          BEGIN(A);
<A>b          BEGIN(B);
<B>a          BEGIN(C);
<B>b          BEGIN(D);
<C>a          BEGIN(A);
<C>b          BEGIN(B);
<D>a          BEGIN(C);
<D>b          BEGIN(D);
<G>a          BEGIN(E);
<G>b          BEGIN(F);
<F>a          BEGIN(G);
<F>b          BEGIN(H);
\n {
   if(YY_START==A || YY_START==B || YY_START==C || YY_START==D)
        printf("Accepted\n");
   else
        printf("Rejected\n");
   BEGIN(INITIAL);
}
. ;
%%
int main(){yylex();}
```

Q12. Identify Integer, Float, and Identifier

```
%{
%}
%x A B C
%%
[a-zA-Z_][a-zA-Z0-9_]* { printf("Identifier: %s\n", yytext); }
[0-9]+\.[0-9]+          { printf("Float: %s\n", yytext); }
[0-9]+                  { printf("Integer: %s\n", yytext); }
[ \t\n]+                ;
.                       { printf("Unknown: %s\n", yytext); }
%%
int main(){yylex();}
```

Q13. YACC/LEX for Language L = {a^n b^n | n>=1}

```
// lex13.l
%%
"a"     return 'a';
"b"     return 'b';
\n      return '\n';
.       return yytext[0];
%%

// yacc13.y
%{
#include <stdio.h>
#include <stdlib.h>
extern int yylex();
extern int yyparse();
void yyerror(const char* s);
%}
%token A B
%%
S: 'a' S 'b'
 | 'a' 'b'
 ;
```

```
str: S '\n' { printf("String accepted\n"); exit(0); }
   ;
%%
void yyerror(const char* s) { fprintf(stderr, "String rejected\n"); }
int main() { yyparse(); return 0; }
```

Q14. YACC/LEX to Recognize Arithmetic Expression

```
// lex14.l
%{
#include "y.tab.h"
%}
%%
[0-9]+      { return NUMBER; }
[+\-*/()]   { return yytext[0]; }
\n          { return '\n'; }
[ \t]       ;
.           { printf("Invalid character %c\n", yytext[0]); }
%%

// yacc14.y
%{
#include <stdio.h>
#include <stdlib.h>
extern int yylex();
void yyerror(const char* s);
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%%
line: expr '\n' { printf("Valid Expression\n"); exit(0); }
    ;
expr: expr '+' expr
    | expr '-' expr
    | expr '*' expr
    | expr '/' expr
    | '(' expr ')'
    | NUMBER
    ;
%%
void yyerror(const char* s) { fprintf(stderr, "Invalid Expression\n"); }
int main() { yyparse(); return 0; }
```

Q15. YACC/LEX to Evaluate Arithmetic Expression

```
// lex15.l
%{
#include "y.tab.h"
%}
%%
[0-9]+      { yylval = atoi(yytext); return NUMBER; }
[+\-*/()\n] { return yytext[0]; }
[ \t]       ;
.           { printf("Invalid character %c\n", yytext[0]); }
%%

// yacc15.y
%{
#include <stdio.h>
#include <stdlib.h>
extern int yylex();
void yyerror(const char* s);
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%%
line: expr '\n' { printf("Result: %d\n", $1); exit(0); }
    ;
expr: expr '+' expr   { $$ = $1 + $3; }
    | expr '-' expr   { $$ = $1 - $3; }
    | expr '*' expr   { $$ = $1 * $3; }
    | expr '/' expr   { if ($3 != 0) $$= $1 / $3; else { yyerror("Divide by zero"); $$ = 0; } }
    | '(' expr ')'    { $$ = $2; }
    | NUMBER          { $$ = $1; }
    ;
%%
void yyerror(const char* s) { fprintf(stderr, "Error: %s\n", s); }
int main() { yyparse(); return 0; }
```