Bangladesh University of Business and Technology (BUBT)

LAB REPORT

Course Title : Compiler Design Lab

Course Code : CSE 324

Submitted By:

Name : Md Jobaer Islam Alif

Id : 20234103391

Intake : 52

Section : 10

Submitted To:

Name : A. S. M. Shafi
(Assistant Professor)

Dept of : CSE
Bangladesh University of Business and Technology

Submission Date: 07/09/25

Signature:_____

Task 1: Recognize and Count Vowels and Consonants

Procedure

Scan characters; increment vowel counter for [AEIOUaeiou], and consonant counter for [A-Za-z] not captured as vowels. Ignore others. After EOF, print counts.

Program (Lex)

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

%%
[AEIOUaeiou]    { vowels++; }
[A-Za-z]        { consonants++; }
.|\n            { /* ignore other chars */ }
%%

int yywrap() {
   return 1;
}
int main() {
   yylex();
   printf("Vowels: %d\nConsonants: %d\n", vowels, consonants);
   return 0;
}
```

Input & Output

```
$ ./a.exe
adasdaSADADDAD
ADFADFAFDafdaf
Vowels: 11
Consonants: 17
```

Task 2: Recognize and Count Uppercase and Lowercase Letters

Procedure

Use character classes: [A-Z] increments uppercase count; [a-z] increments lowercase count. Ignore others and print totals at EOF.

Program (Lex)

```
%{
#include <stdio.h>
#include <ctype.h>
int upperc = 0, lowerc = 0;
%}

%%
[A-Z]    { upperc++; }
[a-z]    { lowerc++; }
./\n     { /* ignore */ }
%%

int yywrap() {
   return 1;
}
int main() {
   yylex();
   printf("\n\nUppercase: %d\nLowercase: %d\n", upperc,
lowerc);
   return 0;
}
```

Input & Output

```
$ ./a.exe
adsaASDADdad
DASDafdsdfAD


Uppercase: 11
Lowercase: 13
```

Task 3: Recognize and Count Tokens from an Input File

Procedure

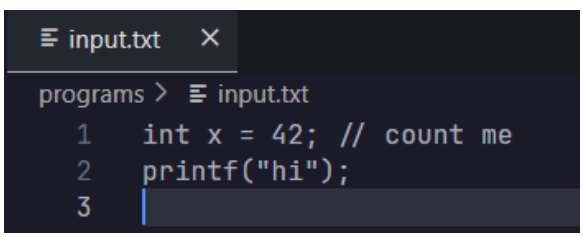
Tokenize input with regex: identifiers, numbers, operators, punctuation, strings. Maintain counters and exclude C keywords from identifiers. Count both // and /* */ comments (per line for block comments). Print all tallies at EOF.

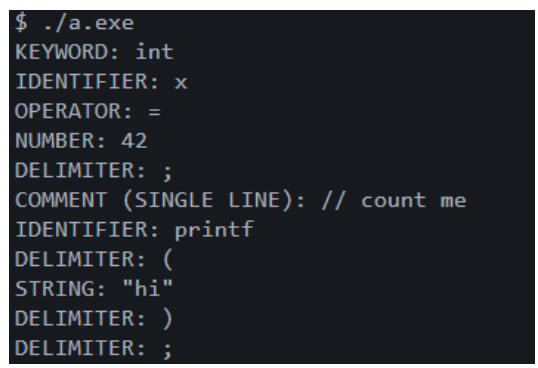
Program (Lex)

```
%{
#include <stdio.h>
#include <string.h>
FILE *yyin;
%}

%%

"int"|"float"|"char"|"if"|"else"|"while"|"return"  {
printf("KEYWORD: %s\n", yytext); }

[0-9]+(\.[0-9]+)?    { printf("NUMBER: %s\n", yytext); }

"=="|"<="|">="|"!="|">"|"<"   { printf("RELATIONAL OPERATOR:
%s\n", yytext); }

"="|"+"|"-"|"*"|"/"   { printf("OPERATOR: %s\n", yytext); }

"("             { printf("DELIMITER: %s\n", yytext); }
")"             { printf("DELIMITER: %s\n", yytext); }
"{"             { printf("DELIMITER: %s\n", yytext); }
"}"             { printf("DELIMITER: %s\n", yytext); }
";"             { printf("DELIMITER: %s\n", yytext); }
","             { printf("DELIMITER: %s\n", yytext); }

\"[^\"]*\"        { printf("STRING: %s\n", yytext); }

"//".*            { printf("COMMENT (SINGLE LINE): %s\n", yytext); }
"/*"([^*]|\*+[^*/])*\*+"/"    { printf("COMMENT (MULTI LINE):
%s\n", yytext); }
[a-zA-Z_][a-zA-Z0-9_]*  { printf("IDENTIFIER: %s\n", yytext); }

[ \t\n]+          ; // Ignore whitespace

.               { printf("UNKNOWN: %s\n", yytext); }
%%
```

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

int main(int argc, char **argv) {
   yyin = fopen("input.txt", "r");
   if (!yyin) {
      perror("Failed to open input.txt");
      return 1;
   }

   yylex();

   fclose(yyin);
   return 0;
}
```

Input & Output





Task 4: Recognize Strings Ending with "aa"

Procedure

Match each line with regex ^.*aa$ to detect lines ending in 'aa'. Print the matched line label; ignore others.

Program (Lex)

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

%%
^.*aa$     { printf("Matched: %s\n", yytext); }
^.*\n      { /* not matched; consume line */ }
.          { /* consume */ }
%%

int yywrap() {
   return 1;
}
int main(){
   yylex();
   return 0;
}
```

Input & Output

```
$ ./a.exe
caa
Matched: caa

aba
baa
Matched: baa
```

Task 5: Recognize Strings Starting and Ending with "bb"

Procedure

Match lines starting with 'bb' and ending with 'bb' using ^bb.*bb$. Print matches; ignore non-matching lines.

Program (Lex)

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

%%
^bb.*bb$   { printf("Matched: %s\n", yytext); }
^.*\n      { /* not matched */ }
.          { /* consume */ }
%%

int yywrap() {
   return 1;
}
int main(void){
   yylex();
   return 0;
}
```

Input & Output

```
$ ./a.exe
bbhellobb
Matched: bbhellobb

bbxxb
startbb
bbend
bbgsfdgbb
Matched: bbgsfdgbb
```

Task 6: Recognize Numbers with '1' in 3rd and '9' in 7th Position

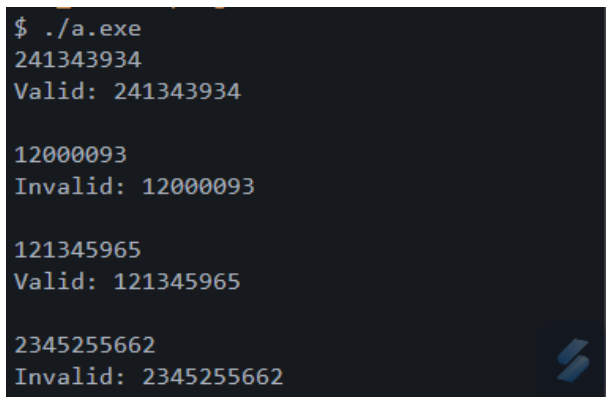
Procedure

Use ^[0-9]{2}1[0-9]{3}9[0-9]*$ to ensure the 3rd digit is '1' and the 7th digit is '9'. Lines that are numeric but fail the pattern are reported 'Invalid'.

Program (Lex)

```
%{
#include <stdio.h>
%}
%%
^[0-9]{2}1[0-9]{3}9[0-9]*$   { printf("Valid: %s\n", yytext); }
^[0-9]+\n?                   { printf("Invalid: %s\n", yytext); }
^.*\n                        { /* ignore non-numeric lines */ }
%%
int main(void){
  yylex();
  return 0;
}
```

Input & Output



Task 7: Count Characters, Words, Spaces, and Lines in a File

Procedure

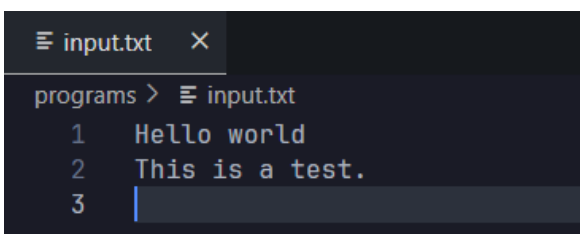
Count: words by matching [^\n]+; spaces by [ ]+; lines by \n; characters by summing token lengths (including spaces and newlines).

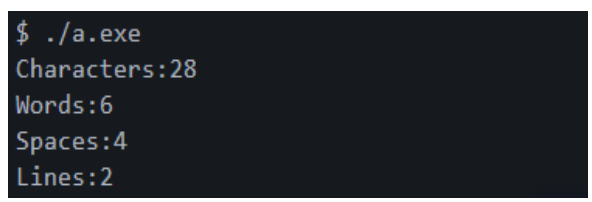
Program (Lex)

```
%{
#include <stdio.h>
#include <string.h>
FILE *yyin;
long chars=0, words=0, spaces=0, lines=0;
%}
```

```
%%
[^\n ]+     { chars += yyleng; words++; }
[ ]+        { chars += yyleng; spaces += yyleng; }
\n          { chars++; lines++; }
%%

int yywrap() {
   return 1;
}
int main(){
   yyin = fopen("input.txt", "r");
   if (!yyin) {
      perror("Failed to open input.txt");
      return 1;
   }
 yylex();
printf("Characters:%ld\nWords:%ld\nSpaces:%ld\nLines:%ld\n",
chars,words,spaces,lines);
 fclose(yyin);
return 0;
}
```

Input & Output





Task 8: Count the Number of Comment Lines in a C Program

Procedure

Count comment lines: //...\n adds 1. For /* ... */ block comments, enter a start condition and increment for each newline encountered within the block (plus the starting line).

Program (Lex)

```
%{
#include <stdio.h>
int c = 0, m = 0;
%}

%%
[/]{1}[/]{1}[a-z A-Z0-9]*        { c++; }
[/]{1}[*]{1}[a-z A-Z0-9]*[*]{1}[/]{1}   { m++; }
.*/\n*                    { ; }
%%

int yywrap() { return 1; }
int main() {
   printf("Enter code (Ctrl+D to stop):\n");
   yylex();
   printf("\n\nComment Lines : %d", c+m);
   return 0;
}
```

Sample Input & Output

```
$ ./a.exe
Enter code (Ctrl+D to stop):
//dasd
//dad
adsad
/*afaf*/


Comment Lines : 3
```

Task 9: Recognize Whether a Sentence is Simple or Compound

Procedure

Heuristic: a sentence containing a semicolon or a comma followed by a coordinating conjunction (for, and, nor, but, or, yet, so) is labeled 'Compound'; otherwise 'Simple'. Process input line by line.

Program (Lex)

```
%{
#include <stdio.h>
#include <string.h>
int compound = 0;
%}
%%
.*;.*\n                    { compound = 1; printf("Compound: %s\n",
yytext); }
.*,([ \t])*((for|and|nor|but|or|yet|so))\b.*\n  { compound = 1;
printf("Compound: %s\n", yytext); }
^.*\n                      {
                          if(!compound) printf("Simple: %s\n", yytext);
                          compound = 0;
                         }
./\n                       { /* consume */ }
%%

int yywrap() { return 1; }
int main(){
  yylex();
  return 0;
}
```

Input & Output

```
$ ./a.exe
I code daily.
Simple: I code daily.

I code, and I learn.
Simple: I code, and I learn.

Study hard; succeed.
Compound: Study hard; succeed.
```

Task 10: Recognize and Count Identifiers

Procedure

Match identifiers with [A-Za-z_][A-Za-z0-9_]* and exclude C keywords from the count. Print total at EOF.

Program (Lex)

```
%{
#include <stdio.h>
#include <string.h>
int id_count = 0;
const char* keywords[] = {

"auto","break","case","char","const","continue","default","do","dou
ble","else","enum",

"extern","float","for","goto","if","inline","int","long","register","rest
rict","return",

"short","signed","sizeof","static","struct","switch","typedef","union"
,"unsigned","void",
   "volatile","while","_Bool","_Complex","_Imaginary", NULL
};
int is_keyword(const char* s){
   for(int i=0; keywords[i]!=NULL; ++i)
if(strcmp(s,keywords[i])==0) return 1;
   return 0;
}
%}
%%
[A-Za-z_][A-Za-z0-9_]*   { if(!is_keyword(yytext)) { id_count++; } }
./\n                     { /* ignore others */ }
%%

int yywrap() { return 1; }
int main(void){
   yylex();
   printf("\n\nIdentifiers: %d\n", id_count);
   return 0;
}
```

Sample Input

```
$ ./a.exe
int main(){ int count=0; float rate; item_id = count + 1; }


Identifiers: 5
```