QUESTION 1:

/\*Lex program to take input from file and remove multiple spaces, newline and tab and write output in a separate file\*/

%{

#include<stdio.h>

%}

%%

[ \n\t]+ {fprintf(yyout, " ");}

. { fprintf(yyout, "%s", yytext); }

%%

int yywrap(){ return 1; }

int main(){

extern FILE \*yyin, \*yyout;

yyin = fopen("input.txt", "r");

yyout = fopen("output.txt", "w");

yylex();

return 0;

}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex remove\_space\_from\_file.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

input.txt

Hello how are you???

output.txt

Hello how are you ???

QUESTION 2

/\*Lex program to take check whether the given number is even or odd 8?\*/

%{

#include<stdio.h>

int i;

%}

%%

[0-9]+ {i=atoi(yytext);

if(i%2==0)

printf("Even");

else

printf("Odd");}

.\* {printf("Not a Number");}

%%

int yywrap(){}

int main()

{

printf("\nEnter the input: ");

yylex();

return 0;

}

Output:

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex remove\_space\_from\_file.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

Enter the input: 5

Odd

6

Even

QUESTION 3

/\*Write a lex code to identify whether a given expression is valid or not.\*/

%{

#include<stdio.h>

#include<stdlib.h>

int c,d,bo=0,bc=0;

%}

%%

[a-zA-Z0-9]+ {c++;}

[+|-|\*] {c--;}

"(" {if(bc<=bo) {bo++;}}

")" {bc++;}

\n {

if((bo==bc) && (c==1)) {printf("Valid Expression\n");}

else {printf("Invalid Expression.\n");}

}

. {}

%%

int yywrap(){ return 1; }

int main(){

printf("Enter expression : \n");

yylex();

return 0;

}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex valid\_arithmetic\_exp.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

Enter expression :

a+b\*c

Valid Expression

a-c\*d+(e-d))

Invalid Expression.

QUESTION 4

/\*lex pgrm to identify and print valid identifiers of C or C++ in a given input pattern.\*/

%{

#include<stdio.h>

int count=0;

%}

%%

auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|return|union|continue|for|signed|void|do|if|static|while|default|goto|sizeof|volatile|const|float|short|unsigned|main;

([a-zA-Z][0-9])+|[a-zA-Z]\* {printf("%s Identifier\n",yytext);count++;}

[0-9][a-zA-Z0-9\_]+ {printf("%s Not a Identifier\n",yytext);}

.|\n|[ ] ;

%%

int yywrap(){return 1;}

int main()

{

extern FILE \*yyin,\*yyout;

yyin=fopen("input4.txt","r");

yyout=fopen("output4.txt","w");

yylex();

printf("Total identifiers: %d\n",count);

return 0;}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex remove\_space\_from\_file.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

a Identifier

b Identifier

c Identifier

f1 Identifier

2f Not a Identifier

Total identifiers: 4

QUESTION 5

/\*Write a Lex program to print integer and float values.\*/

%{

#include<stdio.h>

%}

DIGIT [0-9]

%%

{DIGIT}+ {printf("Integer Value.");}

{DIGIT}+?\.{DIGIT}\* {printf("Float Value.");}

%%

int yywrap(){ return 1; }

int main()

{

yylex();

return 0;

}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex remove\_space\_from\_file.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

45.8

Float Value.

32

Integer Value.

32.

Float Value.

QUESTION 6

/\*Design a lex code to remove the comment section from a C program during the run time.\*/

%{

#include<stdio.h>

%}

%%

\/\/(.\*) {};

\/\\*(.\*\n)\*.\*\\*\/ {};

. {fprintf(yyout, "%s", yytext);}

%%

int yywrap(){ return 1; }

int main() {

extern FILE \*yyin, \*yyout;

yyin=fopen("input6.txt","r");

yyout=fopen("output6.txt","w");

yylex();

return 0;

}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex remove\_comment\_from\_file.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ gcc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

input6.txt

/\*Lex program to take check whether

the given number is even or odd and if neither then chekc for NaN.\*/

%{

#include<stdio.h>

int i;

%}

//Lex Code

output6.txt

%{#include<stdio.h>

int i;%}

QUESTION 7

/\* lex code to extract all the html tags \*/

%{

#include<stdio.h>

%}

%%

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

.|\n {}

%%

int yywrap(){ return 1; }

int main() {

extern FILE \*yyin;

yyin= fopen("input7.txt","r");

yylex();

return 0;

}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex extract\_html\_tags.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

<html>

<head>

<h1>

</h1>

</head>

input7.txt

<html>

<head>

<h1> HELLO </h1>

</head>

QUESTION 8

/\* Design a lex code to identify and print operators, separators, keywords, identifiers. \*/

%{#include<stdio.h>

%}

%%

auto|int|float|char|while|for|if|double|switch|case|long|byte|signed|unsigned|void|const|short|sizeof|return|include|cin|cout {printf("%s KEYWORD\n",yytext);}

\/\/.\* {}

\/\*\.\*(.\*\n)\*.\*\\*\/ {}

[{}(),;] {printf("%s SEPARATOR\n",yytext);}

([a-zA-Z][0-9])+|[a-zA-Z]\* {printf("%s IDENTIFIER\n",yytext);}

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

[\n] {}

[0-9]\* {}

. {}

%%

int yywrap(){}

int main(){

extern FILE \*yyin;

yyin = fopen("input8.txt","r");

yylex();

}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex remove\_space\_from\_file.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

int KEYWORD

a IDENTIFIER

, SEPARATOR

b IDENTIFIER

; SEPARATOR

int KEYWORD

c IDENTIFIER

= OPERATOR

; SEPARATOR

cin KEYWORD

a IDENTIFIER

b IDENTIFIER

; SEPARATOR

c IDENTIFIER

= OPERATOR

a IDENTIFIER

+ OPERATOR

b IDENTIFIER

; SEPARATOR

cout KEYWORD

c IDENTIFIER

; SEPARATOR

return KEYWORD

; SEPARATOR

QUESTION 9

/\*accepts string having even number of a's and odd number of b's \*/

%{

#include<stdio.h>

%}

%%

(aa|bb|(ab|ba)(aa|bb)(ab|ba)) {printf("Accepted");}

.\* {printf("Not Accepted");}

%%

int yywrap(){ return 1; }

int main(){

printf("\nEnter string : ");

yylex();

return 0;

}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex even\_a\_b\_RE.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

Enter string :

ababaabbbbab

Not Accepted

abbabbaa

Accepted

QUESTION 10

/\*Develop a DFA program for the given input string to have 'a' as the third last element.\*/

%{

#include<stdio.h>

%}

%%

(a|b)\*a(aa|bb|ab|ba) printf("The given input string %s is correct.\n",yytext);

.\* printf("%s is invalid.\n",yytext);

%%

int yywrap() { return 1; }

int main() {

printf("Enter the string: \n");

yylex();

return 0;

}

OUTPUT

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ lex last\_third\_char\_a.l

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ cc lex.yy.c

mehul@Ubuntu:~/Desktop/Lex\_Pgrm$ ./a.out

Enter the string:

abbaab

The given input string abbaab is correct.

abbabba

abbabba is invalid.