Day 1:

CSA1473 COMPILER DESIGN FOR SMART COMPUTERS

S.K. LOHIT DEVESHWAR (192211054)

1. Write a LEX program to identify the capital words from the given input.

%{

#include <stdio.h>

%}

%%

[A-Z]+ printf("Capital Word: %s\n", yytext);

.|\n ; // Ignore all other characters

%%

int main(int argc, char \*\*argv) {

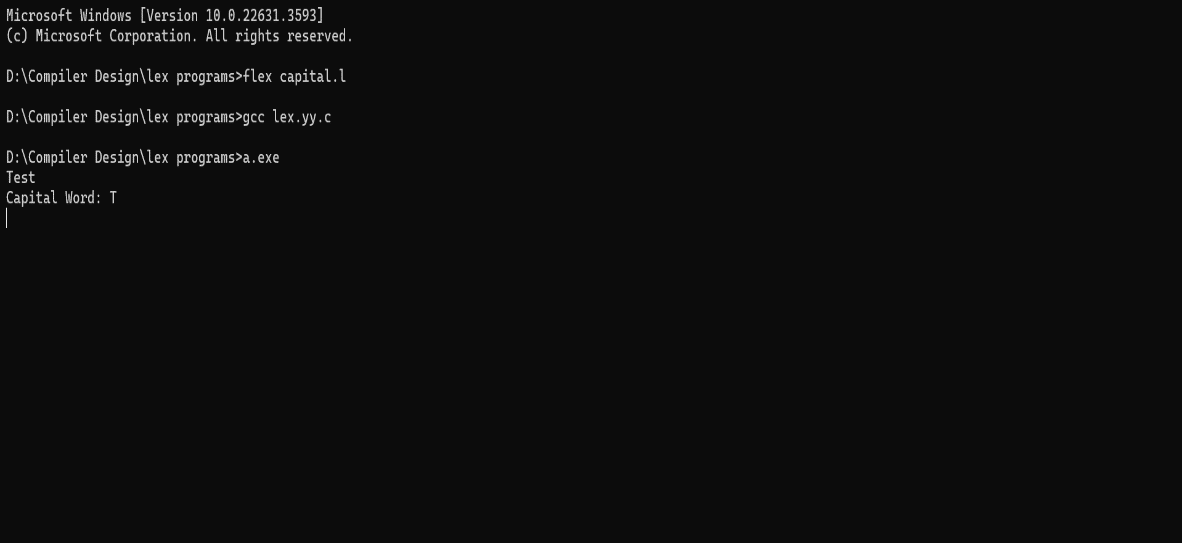
yylex();

return 0;

}

int yywrap() {

return 1;

}

2. Write a LEX program to check whether the given input is digit or not.

%{

#include <stdio.h>

%}

%%

[0-9] printf("Digit: %s\n", yytext); // Print if the character is a digit

[^0-9] printf("Not a digit: %s\n", yytext); // Print if the character is not a digit

%%

int main(int argc, char \*\*argv) {

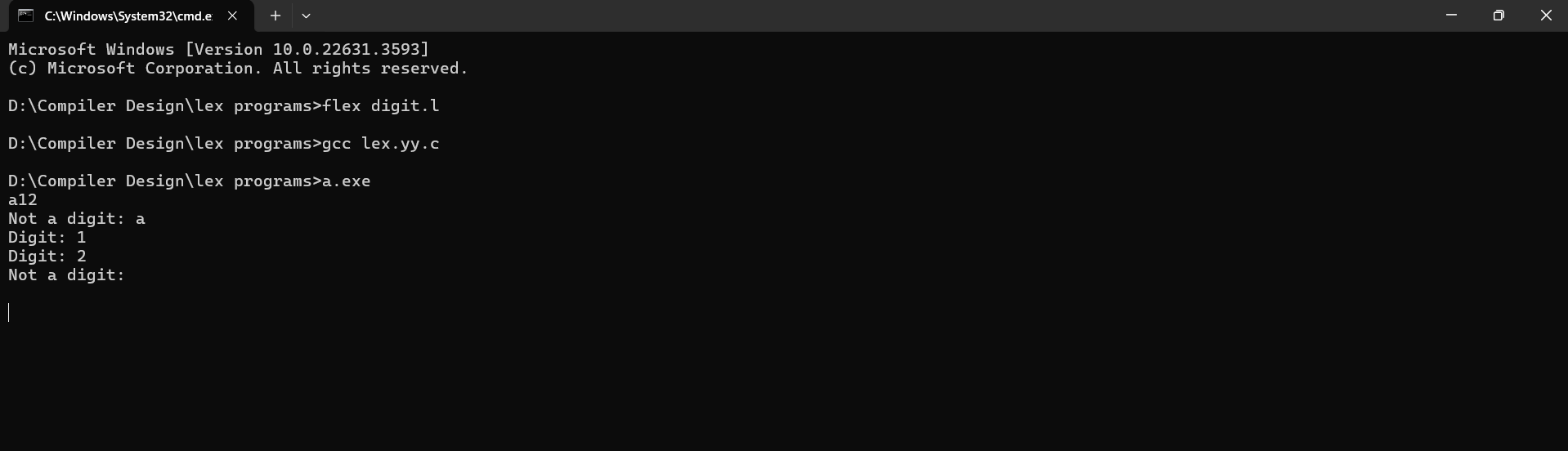
yylex();

return 0;

}

int yywrap() {

return 1;

}

3. The Company ABC runs with employees with several departments. The Organization manager had all the mobile numbers of employees. Assume that you are the manager and need to verify the valid mobile numbers because there may be some invalid numbers present. Implement a LEX program to check whether the mobile number is valid or not.

%{

#include <stdio.h>

%}

%%

^[7-9][0-9]{9}$ { printf("Valid Mobile Number: %s\n", yytext); }

[0-9]{10} { printf("Invalid Mobile Number: %s\n", yytext); }

.|\n ;

%%

int main(int argc, char \*\*argv) {

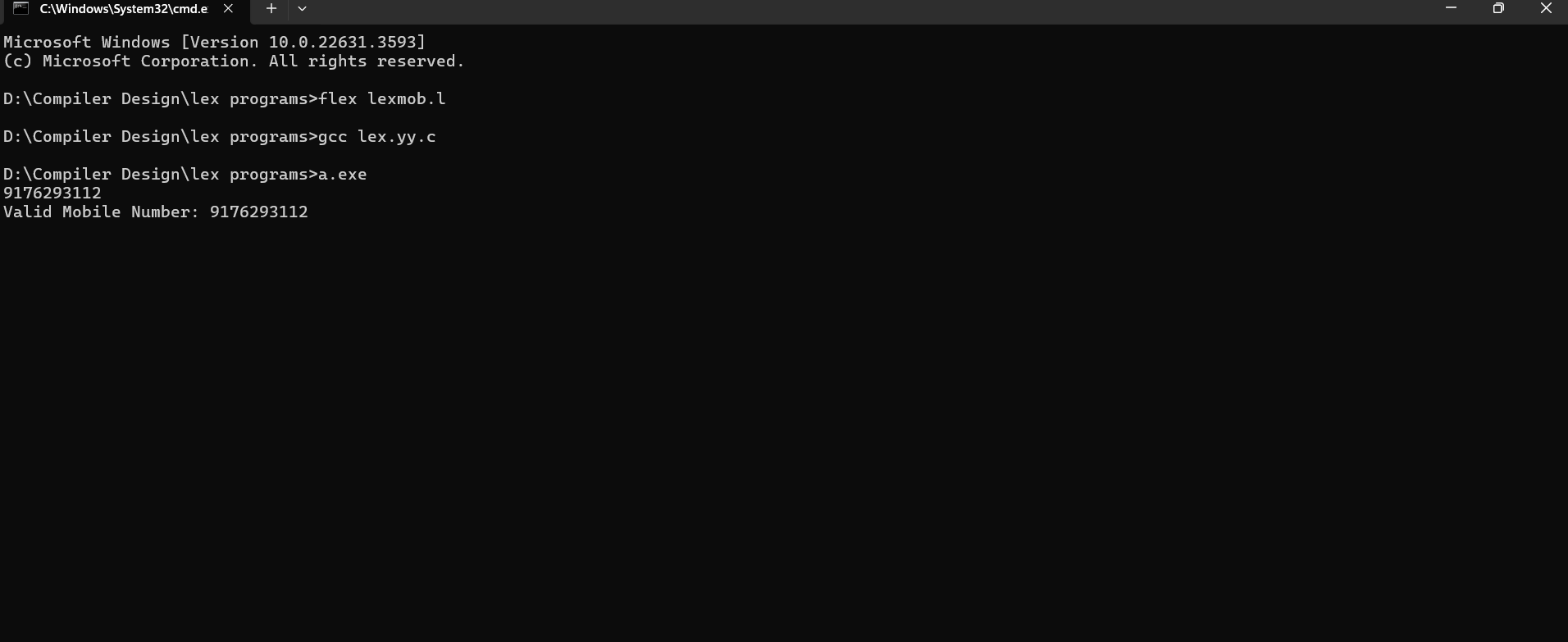
yylex(); // Start the scanning process

return 0;

}

int yywrap() {

return 1;

}

5.Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier. In general there are 32 keywords. The prime function of Lexical Analyser is token Generation. Among the 6 types of tokens, differentiating Keyword and Identifier is a challenging issue. Thus write a LEX program to separate keywords and identifiers.

%{

#include <stdio.h>

#include <string.h>

void print\_token(const char\* token, const char\* type);

%}

%%

"auto"|"break"|"case"|"char"|"const"|"continue"|"default"|"do"|"double"|"else"|"enum"|"extern"|"float"|"for"|"goto"|"if"|"int"|"long"|"register"|"return"|"short"|"signed"|"sizeof"|"static"|"struct"|"switch"|"typedef"|"union"|"unsigned"|"void"|"volatile"|"while" { print\_token(yytext, "Keyword"); }

[a-zA-Z\_][a-zA-Z0-9\_]\* { print\_token(yytext, "Identifier"); }

.|\n ; // Ignore all other characters

%%

void print\_token(const char\* token, const char\* type) {

printf("%s: %s\n", type, token);

}

int main(int argc, char \*\*argv) {

yylex(); // Start the scanning process

return 0;

}

int yywrap() {

return 1;

}



6. Write a LEX program to identify and count positive and negative numbers.

%{

#include <stdio.h>

int positive\_count = 0;

int negative\_count = 0;

%}

%%

-?[1-9][0-9]\* {

if (yytext[0] == '-') {

negative\_count++;

printf("Negative number: %s\n", yytext);

} else {

positive\_count++;

printf("Positive number: %s\n", yytext);

}

}

0 {

positive\_count++;

printf("Positive number: %s\n", yytext);

}

.|\n ; // Ignore all other characters

%%

int main(int argc, char \*\*argv) {

yylex(); // Start the scanning process

printf("Number of positive numbers: %d\n", positive\_count);

printf("Number of negative numbers: %d\n", negative\_count);

return 0;

}

int yywrap() {

return 1;

}

8. Write a LEX program to accept string starting with vowel.

%{

#include <stdio.h>

%}

%%

^[aeiouAEIOU][a-zA-Z]\* { printf("String starting with vowel: %s\n", yytext); }

[a-zA-Z]+ { printf("String not starting with vowel: %s\n", yytext); }

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

.|\n ; // Ignore all other characters

%%

int main(int argc, char \*\*argv) {

yylex(); // Start the scanning process

return 0;

}

int yywrap() {

return 1;

}

