**Experiment 2. Write a program to scan the input program and generate a symbol table.**

%{

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Identifier {

char name[50];

char type[20];

};

struct Identifier id\_table[100];

int id\_count = 0;

void add\_identifier(char \*name, char \*type) {

for (int i = 0; i < id\_count; i++) {

if (strcmp(id\_table[i].name, name) == 0) {

return;

}

}

strcpy(id\_table[id\_count].name, name);

strcpy(id\_table[id\_count].type, type);

id\_count++;

}

void print\_table() {

printf("Identifier Table:\n");

printf("| %-20s | %-10s |\n", "Identifier", "Type");

printf("|----------------------|------------|\n");

for (int i = 0; i < id\_count; i++) {

printf("| %-20s | %-10s |\n", id\_table[i].name, id\_table[i].type);

}

}

void yyerror(const char \*s) {

fprintf(stderr, "Error: %s\n", s);

}

%}

%%

[ \t\n]+ ;

int|float|char|double { add\_identifier(yytext, "Primitive Type"); }

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

[0-9][a-zA-Z0-9\_]\* {

printf("Error: Invalid identifier '%s'. It must start with a letter or an underscore.\n", yytext);

}

[^a-zA-Z0-9\_]+ { }

%%

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

if (argc != 2) {

printf("Usage: %s <filename>\n", argv[0]);

return 1;

}

yyin = fopen(argv[1], "r");

if (!yyin) {

perror("Failed to open file");

return 1;

}

yylex();

print\_table();

return 0;

}

OUTPUT:

