Code:

/\* Declarations \*/

%{

enum {

LOOKUP = 0, /\* default - looking rather than defining. \*/ INT,

FLOAT,

DOUBLE,

DEF

};

int state;

int add\_count = 1000;

int add\_word(int type, char \*word);

int lookup\_word(char \*word);

%}

/\* Rules \*/

%%

[\n] { state = LOOKUP; }

/\* end of line, return to default state \*/

/\* whenever a line starts with a reserved keyword \*/ /\* start defining identifiers of that type \*/

[(] { state = LOOKUP; }

^int { state = INT; }

^float { state = FLOAT; }

^double { state = DOUBLE; }

^def { state = DEF; }

end {return 0;}

[0-9][a-zA-Z0-9\_]\* /\* ignore invalid identifiers \*/; [a-zA-Z\_][a-zA-Z0-9\_]\*/[(] {

if(state != LOOKUP) {

/\* define the current lexeme \*/add\_word(DEF, yytext); }

}

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

/\* identify lexeme, define it or look it up \*/

if(state != LOOKUP) {

/\* define the current lexeme \*/

add\_word(state, yytext);

}

}

. /\* ignore anything else \*/ ;

%%

/\*\*\* Main code \*\*\*/

int yywrap(){}

struct word \*word\_list; /\* first element in word list \*/

/\* define a linked list of words, their types, values and addresses \*/

struct word {

char \*word\_name;

int word\_type;

int word\_address;

struct word \*next;

};

extern void \*malloc() ;

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

{

yylex();

struct word \*wp = word\_list;

printf("<--------------------Symbol Table-------------------->\nIdentifier\t\tAddress\t\tType\n"); /\* display entire list i.e symbol table \*/

for(; wp; wp = wp->next) {

switch(wp->word\_type) {case INT: printf("%s\t\t\t%d\t\tInteger\n", wp->word\_name, wp->word\_address); break;

case FLOAT: printf("%s\t\t\t%d\t\tFloat\n", wp->word\_name, wp->word\_address); break; case DOUBLE: printf("%s\t\t\t%d\t\tDouble\n", wp->word\_name, wp->word\_address); break; case DEF: printf("%s\t\t%d\t\tFunction\n", wp->word\_name, wp->word\_address); break; }

}

return 0;

}

int add\_word(int type, char \*word)

{

struct word \*wp;

if(lookup\_word(word) != LOOKUP) {

printf("!!! warning: identifier %s already defined \n", word);

return 0;

}

/\* identifier not there, allocate a new entry and link it on the list \*/

wp = (struct word \*) malloc(sizeof(struct word));

wp->next = word\_list;

/\* have to copy the indentifier itself as well \*/

wp->word\_name = (char \*) malloc(strlen(word)+1);

strcpy(wp->word\_name, word);

wp->word\_type = type;

add\_count += 4;

wp->word\_address = add\_count;

word\_list = wp;

return 1;

}

int lookup\_word(char \*word)

{

struct word \*wp = word\_list;

/\* search down the list looking for the word \*/for(; wp; wp = wp->next) {

if(strcmp(wp->word\_name, word) == 0)

return wp->word\_type;

}

return LOOKUP; /\* not found \*/

}

