

ASSIGNMENT 3 WEEK-2

Name:Pranav Sarda

Gr No: 21810773

Roll No: 321047

Div: A-2

3 B :-

```
%{
/*
 * Word recognizer with a symbol table.
 */
enum{
    LOOKUP = 0, /* default-looking rather than defining. */
    VERB,
    ADJ,
    ADV,
    NOUN,
    PREP,
    PRON,
    CONJ,
    INT
};
int state;
int add_word(int type, char *word);
int lookup_word(char *word);
}%
%%
\n    {state = LOOKUP; }
    /* whenever a line starts with a reserved part of speech name */
    /* start defining words of that type */
^verb { state = VERB; }
^adj  { state = ADJ; }
^adv  { state = ADV; }
^noun { state = NOUN; }
^prep { state = PREP; }
^pron { state = PRON; }
^conj { state = CONJ; }
^int  { state = INT; }
/* a normal word, define it or look it up */
[a-zA-Z]+ {
    if(state != LOOKUP) {
        /* define the current word */
        add_word(state, yytext);
    } else {
        switch(lookup_word(yytext)) {
            case VERB: printf("%s: verb\n", yytext); break;
            case ADJ:  printf("%s: adjective\n", yytext); break;
            case ADV:  printf("%s: adverb\n", yytext); break;
            case NOUN: printf("%s: noun\n", yytext); break;
```

ASSIGNMENT 3 WEEK-2

```
        case PREP: printf("%s: preposition\n", yytext); break;
        case PRON: printf("%s: pronoun\n", yytext); break;
        case CONJ: printf("%s: conjunction\n", yytext); break;
        case INT: printf("%s: interjection\n", yytext); break;
        default:
            printf("%s: don't recognize\n", yytext) ;
            break;
    }
}
}
/* ignore anything else */;
%%
int yywrap(void)
{
}
int main()
{
    yylex();
}
/* define a linked list of words and types */
struct word{
    char *word_name;
    int word_type;
    struct word *next;
};
struct word *word_list; /* first element in word list */
extern void *malloc();
int add_word(int type,char *word)
{
    struct word *wp;
    if(lookup_word(word) != LOOKUP) {
        printf (" ! ! ! warning: word %s already defined \n",word);
        return 0;
    }
    /* word 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 word itself as well */
    wp->word_name = (char *) malloc(strlen(word)+1);
    strcpy(wp->word_name,word);
    wp->word_type = type;
    word_list = wp;
    return 1; /* it worked */
}
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 */
}
```

ASSIGNMENT 3 WEEK-2

OUTPUT :-

```
Applications ▾ Places ▾ Terminal ▾ Sat 03:10
root@kali: ~/Documents/LPCC/3rd_assign/3b
File Edit View Search Terminal Help
root@kali:~# cd Documents/LPCC/3rd_assign/3b
root@kali:~/Documents/LPCC/3rd_assign/3b# lex 3b.l
root@kali:~/Documents/LPCC/3rd_assign/3b# gcc lex.yy.c
root@kali:~/Documents/LPCC/3rd_assign/3b# ./a.out
verb is am
is
is: verb
prep in of below
noun center boy girl ring boxers punches champ
verb touch trade goes
adj two exhausted
adv very quickly tentatively down
conj if then and until
pron I we she
int Oh! Wow!
In the center of the ring, the two exhausted boxers touch gloves and tentatively trade punches until Oh! down goes the champ
In: preposition
the: don't recognize
center: noun
of: preposition
the: don't recognize
ring: noun
two: adjective
exhausted: adjective
boxers: noun
touch: verb
gloves: don't recognize
and: conjunction
tentatively: adverb
trade: verb
punches: noun
until: conjunction
Oh: interjection
down: adverb
goes: verb
the: don't recognize
champ: noun
[]
```

ASSIGNMENT 3 WEEK-2

3D :-

```
%{
/*
 * Word recognizer with a symbol table.
 */
enum {
    LOOKUP = 0, /* default - looking rather than defining. */
    KEYWORD,
    IDENTIFIER,
    NUMBER,
    LOGICAL,
    ARITHMETIC,
    RELATIONAL,
    FORMAT,
    PUNC,
    ASSIGN
};
int state;
int add_word(int type, char *word);
int lookup_word(char *word);
}%
%%
\n    { state = LOOKUP; } /* end of line, return to default state */
/* whenever a line starts with a reserved part of speech name */
/* start defining words of that type */
^keyword { state = KEYWORD; }
^identifier { state = IDENTIFIER; }
^number { state = NUMBER; }
^logical { state = LOGICAL; }
^arithmetic { state = ARITHMETIC; }
^relational { state = RELATIONAL; }
^format { state = FORMAT; }
^punc { state = PUNC; }
^assign { state = ASSIGN; }
[a-zA-Z0-9\.\&&\\|\\+\\-\\/%\\*\\<\\>==\\<=\\>=\\d%s%f%s\\{\\}\\,\\;\\'\\:\\\"\\?\\!\\)]+ {
    /* a normal word, define it or look it up */
    if(state != LOOKUP) {
        /* define the current word */
        add_word(state, yytext);
    } else {
        switch(lookup_word(yytext)) {
            case KEYWORD: printf("%s: keyword\n", yytext); break;
            case IDENTIFIER: printf("%s: identifier\n", yytext); break;
            case NUMBER: printf("%s: number\n", yytext); break;
            case LOGICAL: printf("%s: Logical operator\n", yytext); break;
            case ARITHMETIC: printf("%s: Aritmetic operator\n", yytext); break;
            case RELATIONAL: printf("%s: Relational operator\n", yytext); break;
            case FORMAT: printf("%s: Format Specifier\n", yytext); break;
            case PUNC: printf("%s: Punctuation Symbol\n", yytext); break;
            case ASSIGN: printf("%s: Assignment Op\n", yytext); break;
            default:
                printf("%s: didn't recognize\n", yytext);
                break;
        }
    }
}
```

ASSIGNMENT 3 WEEK-2

```
    }
    . /* ignore anything else */ ;
%%
int yywrap(void)
{

}
int main()
{
    yylex();
}
/* define a linked list of words and types */
struct word {
    char *word_name;
    int word_type;
    struct word *next;
};
struct word *word_list; /* first element in word list */
extern void *malloc() ;
int
add_word(int type, char *word)
{
    struct word *wp;
    if(lookup_word(word) != LOOKUP) {
        printf("!!! warning: word %s already defined \n", word);
        return 0;
    }
    /* word 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 word itself as well */
    wp->word_name = (char *) malloc(strlen(word)+1);
    strcpy(wp->word_name, word);
    wp->word_type = type;
    word_list = wp;
    return 1; /* it worked */
}
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 */
}
```

ASSIGNMENT 3 WEEK-2

OUTPUT :-

```
File Edit View Search Terminal Help
root@kali:~/Documents/LPCC/3rd_assign/3d# lex 3d.l
root@kali:~/Documents/LPCC/3rd_assign/3d# gcc lex.yy.c
root@kali:~/Documents/LPCC/3rd_assign/3d# ./a.out
keyword for if return else
identifier num al search
logical || &&
arithmetic + - * /
relational < > <= >=
format %u %s %d
punc , ; : ( )
assign =
number 10 20 30
num = 20 ;
num: identifier
=: Assignment Op
20: number
;: Punctuation Symbol
if ( num < search )
if: keyword
(: Punctuation Symbol
num: identifier
<: Relational operator
search: identifier
): Punctuation Symbol
%s
%s: Format Specifier
al = num + 10 ;
al: identifier
=: Assignment Op
num: identifier
+: Arithmetic operator
10: number
:: Punctuation Symbol

```