

**Ex No: 6**

**Date:**

## **RECOGNIZE A VALID VARIABLE WITH LETTERS AND DIGITS USING LEX AND YACC**

**AIM:**

To recognize a valid variable which starts with a letter followed by any number of letters or digits.

**ALGORITHM:**

- Define lexical rules in variable.l with regex to match valid variables: start with a letter, followed by letters or digits. Tokenize input, distinguishing letters and digits.
- Use lexer (variable.l) to tokenize input into meaningful units like letters and digits.
- Implement grammar rules in parser (variable.y) for recognizing valid variable names using context-free grammar. Incorporate lexer tokens into parsing.
- In parser, implement error handling to detect invalid variable names. Set a flag (e.g., valid) to mark invalid identifiers.
- Check validity post-parsing; if flag remains true, indicate valid identifier. Otherwise, display message for invalid input.

**PROGRAM:**

**variable.l:**

```
% {
    #include "y.tab.h"
% }
%%
[a-zA-Z_][a-zA-Z_0-9]* return letter;
[0-9]          return digit;
.              return yytext[0];
\n            return 0;
%%
int yywrap()
{
return 1;
}
```

**variable.y:**

```
% {
    #include <stdio.h>
    int valid=1;
% }
```

```

%token digit letter
%%
start : letter s
s :    letter s
      | digit s |;

%%
int yyerror()
{
    printf("\nIts not a identifier!\n");
    valid=0;
    return 0;
}
int main() {
    printf("\nEnter a name to test for an identifier: ");
    yyparse();
    if(valid) {
        printf("\nIt is a identifier!\n");
    } }

```

### OUTPUT:

```

[root@localhost-live liveuser]# vi 523_ex6.l
[root@localhost-live liveuser]# vi 523_ex6.y
[root@localhost-live liveuser]# lex 523_ex6.l
[root@localhost-live liveuser]# yacc -d 523_ex6.y
[root@localhost-live liveuser]# cc lex.yy.c y.tab.c
[root@localhost-live liveuser]# ./a.out

Enter a name to test for an identifier: var

It is a identifier!
[root@localhost-live liveuser]# ./a.out

Enter a name to test for an identifier: 2

Its not a identifier!

```

### RESULT: