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:

- Include necessary headers and declarations within `%{ %}` in the lexer file.
- Define rules to match identifiers (starting with a letter or underscore, followed by letters, digits, or underscores) and return token `letter`.
- Define a rule to match digits (single digit) and return token `digit`.
- Define a rule to match any other character and return it.
- Define a rule to match newline character and return 0 to indicate end of input.
- Implement `yywrap()` function to return 1, indicating end of input.
- In the parser file, include necessary headers and declarations within `%{ %}`.
- Define tokens `digit` and `letter`.
- Specify grammar rules for parsing identifiers recursively.
- Implement `yyerror()` function to handle parsing errors, setting `valid` flag to 0.
- In `main()` function, prompt the user to enter a name to test for an identifier.
- Call `yyparse()` to initiate parsing.
- If `valid` flag is set, print "It is an identifier", else print "It is not an identifier".

PROGRAM: variable.l:

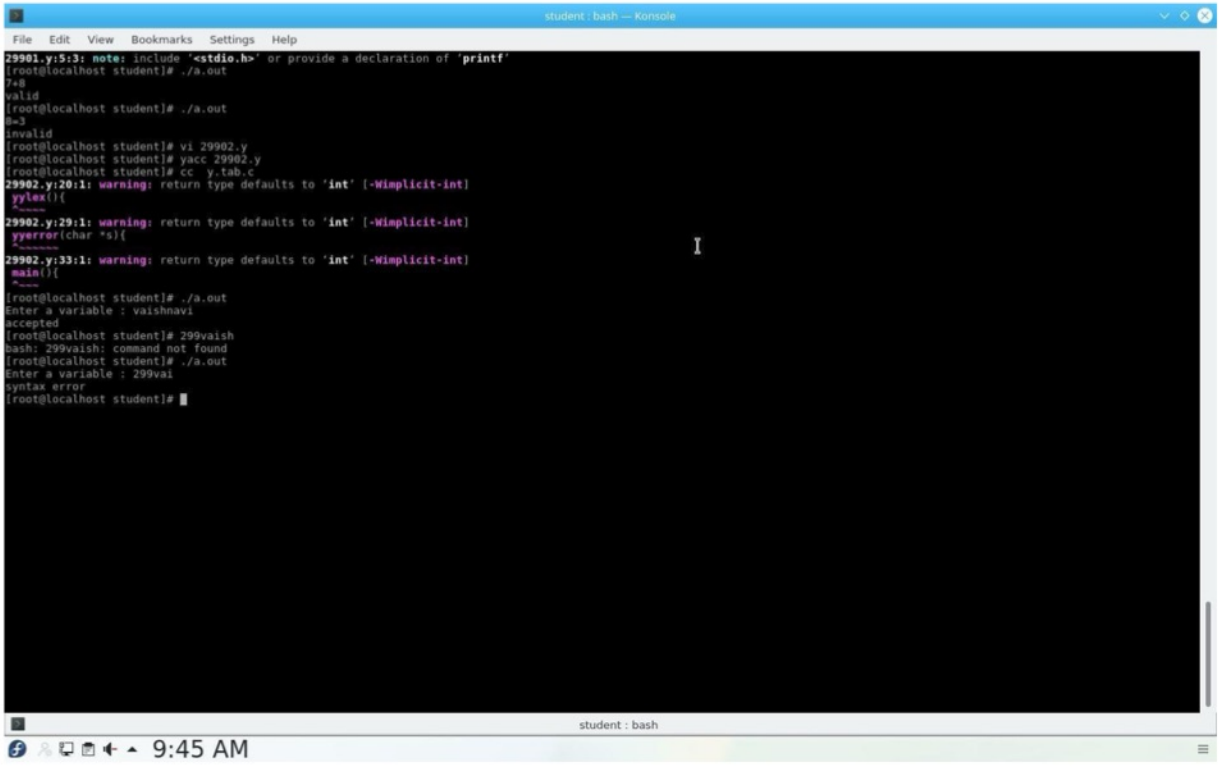
```
%{
   #include "y.tab.h"
%}
%%
[a-zA-Z_][a-zA-Z_0-9]* return letter;
[0-9]                  return digit;
.                   return yytext[0];
```

```
\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 an identifier!\n");     valid=0;
return 0; } int main() {
   printf("\nEnter a name to test for an identifier: ");     yyparse();
if(valid) {
       printf("\nIt is an identifier!\n");
   } }
```

OUTPUT:



RESULT: