BUBT
Committed to Academic Excellence
BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Lab Assessment

Course Code: CSE 324

Course Title: Compiler Design

Submitted to:	Submitted by:
Name: Md. Mamun Hossain	Name: Syeda Nowshin Ibnat
Lecturer	ID: 17183103020
Dept. of CSE	Intake: 39
at Bangladesh University of Business and Technology.	Section: 01
	Program: B.Sc. in CSE
	Semester: Fall 20-21

Date of Submission: 22/03/2021

3 No Answer

Problem name:

Write a program to simulate lexical analyzer for validating operators. Take input multi time.

Sample Code:

```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// Returns 'true' if the character is a DELIMITER.
bool isDelimiter(char ch) {
if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
ch == '[' || ch == ']' || ch == '{' || ch == '}')
return (true);
return (false); }
// Returns 'true' if the character is an OPERATOR.
bool isOperator(char ch) {
if (ch == '+' || ch == '-' || ch == '*' ||
ch == '/' || ch == '>' || ch == '<' ||
ch == '=' || ch == '&&' || ch == '==')
return (true);
return (false); }
// Returns 'true' if the string is a VALID IDENTIFIER.
bool validIdentifier(char* str) {
if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
str[0] == '3' || str[0] == '4' || str[0] == '5' ||
str[0] == '6' || str[0] == '7' || str[0] == '8' ||
str[0] == '9' || isDelimiter(str[0]) == true)
return (false);
return (true); }
// Returns 'true' if the string is a KEYWORD.
bool isKeyword(char* str) {
if (!strcmp(str, "if") || !strcmp(str, "else") ||
!strcmp(str, "while") || !strcmp(str, "do") ||
!strcmp(str, "break") ||
!strcmp(str, "continue") || !strcmp(str, "int")
|| !strcmp(str, "double") || !strcmp(str, "float")
|| !strcmp(str, "return") || !strcmp(str, "char")
|| !strcmp(str, "case") || !strcmp(str, "char")
|| !strcmp(str, "sizeof") || !strcmp(str, "long")
|| !strcmp(str, "short") || !strcmp(str, "typedef")
|| !strcmp(str, "switch") || !strcmp(str, "unsigned")
|| !strcmp(str, "void") || !strcmp(str, "static")
|| !strcmp(str, "struct") || !strcmp(str, "goto"))
return (true);
return (false); }
// Returns 'true' if the string is an INTEGER.
bool isInteger(char* str) {
int i, len = strlen(str);
if (len == 0)
return (false);
for (i = 0; i < len; i++) {
if (str[i] != '0' && str[i] != '1' && str[i] != '2'
&& str[i] != '3' && str[i] != '4' && str[i] != '5'
&& str[i] != '6' && str[i] != '7' && str[i] != '8'
&& str[i] != '9' || (str[i] == '-' && i > 0))
return (false); }
return (true); }
// Returns 'true' if the string is a REAL NUMBER.
bool isRealNumber(char* str) {
int i, len = strlen(str);
bool hasDecimal = false;
if (len == 0)
return (false);
for (i = 0; i < len; i++) {
if (str[i] != '0' && str[i] != '1' && str[i] != '2'
&& str[i] != '3' && str[i] != '4' && str[i] != '5'
&& str[i] != '6' && str[i] != '7' && str[i] != '8'
&& str[i] != '9' && str[i] != '.' ||
(str[i] == '-' && i > 0))
return (false);
if (str[i] == '.')
hasDecimal = true; }
return (hasDecimal); }
// Extracts the SUBSTRING.
char* subString(char* str, int left, int right) {
int i;
char* subStr = (char*)malloc(
sizeof(char) * (right - left + 2));
for (i = left; i <= right; i++)
subStr[i - left] = str[i];
```

```
subStr[right - left + 1] = '\0';
return (subStr); }
// Parsing the input STRING.
void parse(char* str) {
int left = 0, right = 0;
int len = strlen(str);
while (right <= len && left <= right) {
if (isDelimiter(str[right]) == false)
right++;
if (isDelimiter(str[right]) == true && left == right) {
if (isOperator(str[right]) == true)
printf("'%c' IS AN OPERATOR\n", str[right]);
right++;
left = right;
} else if (isDelimiter(str[right]) == true && left != right
|| (right == len && left != right)) {
char* subStr = subString(str, left, right - 1);
if (isKeyword(subStr) == true)
printf("'%s' IS A KEYWORD\n", subStr);
else if (isInteger(subStr) == true)
printf("'%s' IS AN INTEGER\n", subStr);
else if (isRealNumber(subStr) == true)
printf("'%s' IS A REAL NUMBER\n", subStr);
else if (validIdentifier(subStr) == true
&& isDelimiter(str[right - 1]) == false)
printf("'%s' IS A VALID IDENTIFIER\n", subStr);
else if (validIdentifier(subStr) == false
&& isDelimiter(str[right - 1]) == false)
```

printf("'%s' IS NOT A VALID IDENTIFIER\n", subStr);

left = right; } }

return; }

// DRIVER FUNCTION

int main() {

// maximum length of string is 100 here

char str[100] = "int y = a+b;y++; if(x==m && x==n) y=y-1;";

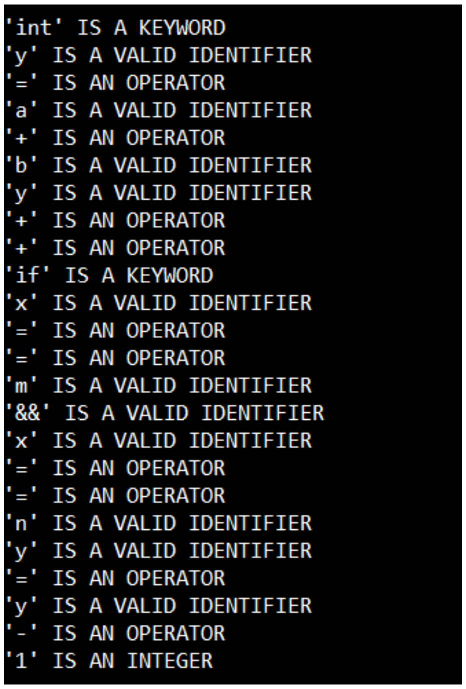
parse(str); // calling the parse function

return (0); }

1) **Sample Input:**

int y = a+b;y++; if(x==m && x==n) y=y-1;

Sample Output:



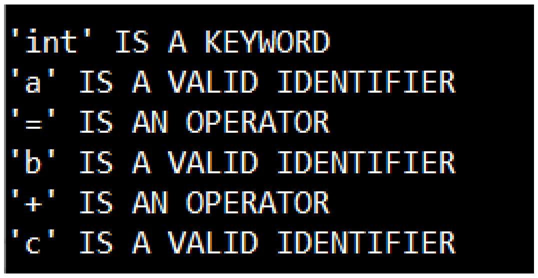
```
'int' IS A KEYWORD
'y' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'a' IS A VALID IDENTIFIER
'+' IS AN OPERATOR
'b' IS A VALID IDENTIFIER
'y' IS A VALID IDENTIFIER
'+' IS AN OPERATOR
'+' IS AN OPERATOR
'if' IS A KEYWORD
'x' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'=' IS AN OPERATOR
'm' IS A VALID IDENTIFIER
'&&' IS A VALID IDENTIFIER
'x' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'=' IS AN OPERATOR
'n' IS A VALID IDENTIFIER
'y' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'y' IS A VALID IDENTIFIER
'-' IS AN OPERATOR
'1' IS AN INTEGER
```

Note: Output is not right for && operator. Otherwise the code is ok.

2) Sample Input:

int a=b+c;

Sample Output:

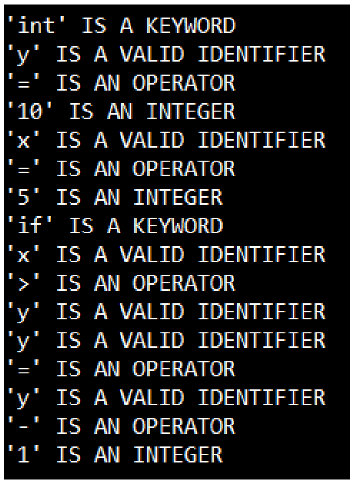


```
'int' IS A KEYWORD
'a' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'b' IS A VALID IDENTIFIER
'+' IS AN OPERATOR
'c' IS A VALID IDENTIFIER
```

3) Sample Input:

int y = 10; x = 5; if(x>y) y=y-1;

Sample Output:



```
'int' IS A KEYWORD
'y' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'10' IS AN INTEGER
'x' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'5' IS AN INTEGER
'if' IS A KEYWORD
'x' IS A VALID IDENTIFIER
'>' IS AN OPERATOR
'y' IS A VALID IDENTIFIER
'y' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'y' IS A VALID IDENTIFIER
'-' IS AN OPERATOR
'1' IS AN INTEGER
```

4 No Answer

Problem Name:

Write a program to find FIRST value of the given grammar. Take the optimized input as 2D array.

Solution:

Given grammar,

S → A

A → aB | Ad $\longrightarrow$

B → b

C → ε

Left Recursion

S → A

A → aBA’ $\longrightarrow$

A’ → dA’ | ε

B → b

C → ε

S=A

A=aBX

X=dX

X=#

B=b

C=#

Here,

→ is =

A’ is X

ε is #

No substation and no left factoring.

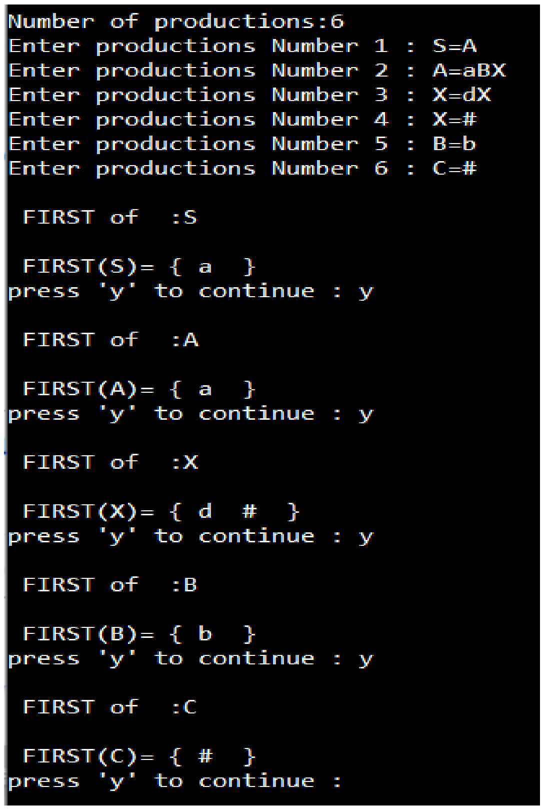
First set
S = {a}
A = {a}
A’ = {d, ε}
B = {b}
C = { ε}

Sample Code:

```
#include<stdio.h>
#include<ctype.h>
void FIRST(char[],char );
void addToResultSet(char[],char);
int numOfProductions;   //limit
char productionSet[6][5];  //product array
int main() {
int i;
char choice;
char c;
char result[30];  // result array
printf("Number of productions:");
scanf(" %d",&numOfProductions);
for(i=0;i<numOfProductions;i++)//read production string eg: E=E+T {
printf("Enter productions Number %d : ",i+1);
scanf(" %s",productionSet[i]); //this will only increment row. }
do {
printf("\n FIRST of  :");
scanf(" %c",&c);
FIRST(result,c); //Compute FIRST; Get Answer in 'result' array
printf("\n FIRST(%c)= {",c);
for(i=0;result[i]!='\0';i++)
printf(" %c ",result[i]);       //Display result
printf(" }\n");
printf("press 'y' to continue : ");
scanf(" %c",&choice); }
while(choice=='y'||choice =='Y'); }
```

```
//Compute the elements in FIRST(c) and write them in Result Array.
void FIRST(char* Result,char c) {
int i,j,k;
char subResult[20];
int foundEpsilon;
subResult[0]='\0';
Result[0]='\0';
//If X is terminal, FIRST(X) = {X}.
if(!(isupper(c))) {
addToResultSet(Result,c);
return ; }
//If X is non terminal
//Read each production
for(i=0;i<numOfProductions;i++) {
if(productionSet[i][0]==c) {
//If X → ε is a production, then add ε to FIRST(X).
if(productionSet[i][2]=='#') addToResultSet(Result,'#');
else {
j=2;
while(productionSet[i][j]!='\0') {
foundEpsilon=0;
FIRST(subResult,productionSet[i][j]);
for(k=0;subResult[k]!='\0';k++)
addToResultSet(Result,subResult[k]);
for(k=0;subResult[k]!='\0';k++)
if(subResult[k]=='#') {
foundEpsilon=1;
break; }
//No ε found, no need to check next element
if(!foundEpsilon)
break;
j++;  }}} }
return ; }
void addToResultSet(char Result[],char val) {
int k;
for(k=0 ;Result[k]!='\0';k++)
if(Result[k]==val)
return;
Result[k]=val;
Result[k+1]='\0'; }
```

Sample Output:



```
Number of productions:6
Enter productions Number 1 : S=A
Enter productions Number 2 : A=aBX
Enter productions Number 3 : X=dX
Enter productions Number 4 : X=#
Enter productions Number 5 : B=b
Enter productions Number 6 : C=#

 FIRST of  :S

 FIRST(S)= { a  }
press 'y' to continue : y

 FIRST of  :A

 FIRST(A)= { a  }
press 'y' to continue : y

 FIRST of  :X

 FIRST(X)= { d  #  }
press 'y' to continue : y

 FIRST of  :B

 FIRST(B)= { b  }
press 'y' to continue : y

 FIRST of  :C

 FIRST(C)= { #  }
press 'y' to continue :
```