Date: 27/09/2023

EXPERIMENT-3.1

Aim:

Write a program to check whether a string include Keyword or not.

Program:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string input;
   cout << "Enter a string: ";
   cin >> input;

   string keywords[] = {
      "auto", "break", "case", "char", "const", "continue", "default", "do",
      "double", "else", "enum", "extern", "float", "for", "goto", "if", "int",
      "long", "register", "return", "short", "signed", "sizeof", "static", "struct",
      "switch", "typedef", "union", "unsigned", "void", "volatile", "while"
   };

   bool isKeyword = false;

   for (const string& keyword : keywords) {
      if (input == keyword) {
         isKeyword = true;
         break;
      }
   }

   if (isKeyword) {
      cout << input << " is a C++ keyword." << endl;
   } else {
      cout << input << " is not a C++ keyword." << endl;
   }
   return 0;
}
```

Output:

```
Enter a string: do
do is a C++ keyword.
```

```
Enter a string: hi
hi is not a C++ keyword.
```

Date: 27/09/2023

EXPERIMENT-3.2

Aim:

Write a program to remove left Recursion from a Grammar.

Program:

```
#include<iostream>

#include<string>

using namespace std;

int main() {
   string ip,op1,op2,temp;
   int sizes[10] = {};
   char c;
   int n,j,l;
   cout<<"Enter the Parent Non-Terminal : ";
   cin>>c;
   ip.push_back(c);
   op1 += ip + "\'->";
   ip += "->";
   op2+=ip;
   cout<<"Enter the number of productions : ";
   cin>>n;
   for(int i=0;i<n;i++) {
      cout<<"Enter Production "<<i+1<<" : ";
      cin>>temp;
      sizes[i] = temp.size();
      ip+=temp;
      if(i!=n-1)
         ip += "|";
   }
   cout<<"Production Rule : "<<ip<<endl;
   for(int i=0,k=3;i<n;i++) {
      if(ip[0] == ip[k]) {
         cout<<"Production "<<i+1<<" has left recursion."<<endl;
         if(ip[k] != '#') {
            for(l=k+1;l<k+sizes[i];l++) {
               op1.push_back(ip[l]);
            }
            k=l+1;
            op1.push_back(ip[0]);
            op1 += "\'|";
         }
      }
```

```
        else {
            cout<<"Production "<<i+1<<" does not have left recursion."<<endl;
            if(ip[k] != '#') {
                for(j=k;j<k+sizes[i];j++) {
                    op2.push_back(ip[j]);
                }
                k=j+1;
                op2.push_back(ip[0]);
                op2 += "\'|";
            }
            else {
                op2.push_back(ip[0]);
                op2 += "\'";
            }
        }
    }
    op1 += "#";
    cout<<op2<<endl;
    cout<<op1<<endl;
    return 0;
}
```

Output:

```
Enter the Parent Non-Terminal : S
Enter the number of productions : 2
Enter Production 1 : Sa
Enter Production 2 : bS
Production Rule : S->Sa|bS
Production 1 has left recursion.
Production 2 does not have left recursion.
S->bSS'|
S'->aS'|#
```