IMPLEMENTATION OF LEFT RECURSION ELIMINATION

NAME: **RISHAL RAMESH** ***EXP NO:*** **4a**

REG NO: **RA1911030010084**

DATE: **10/02/2022**

AIM:

To study and implement left recursion elimination for the given productions.

LANGUAGE USED:

C++

ALGORITHM:

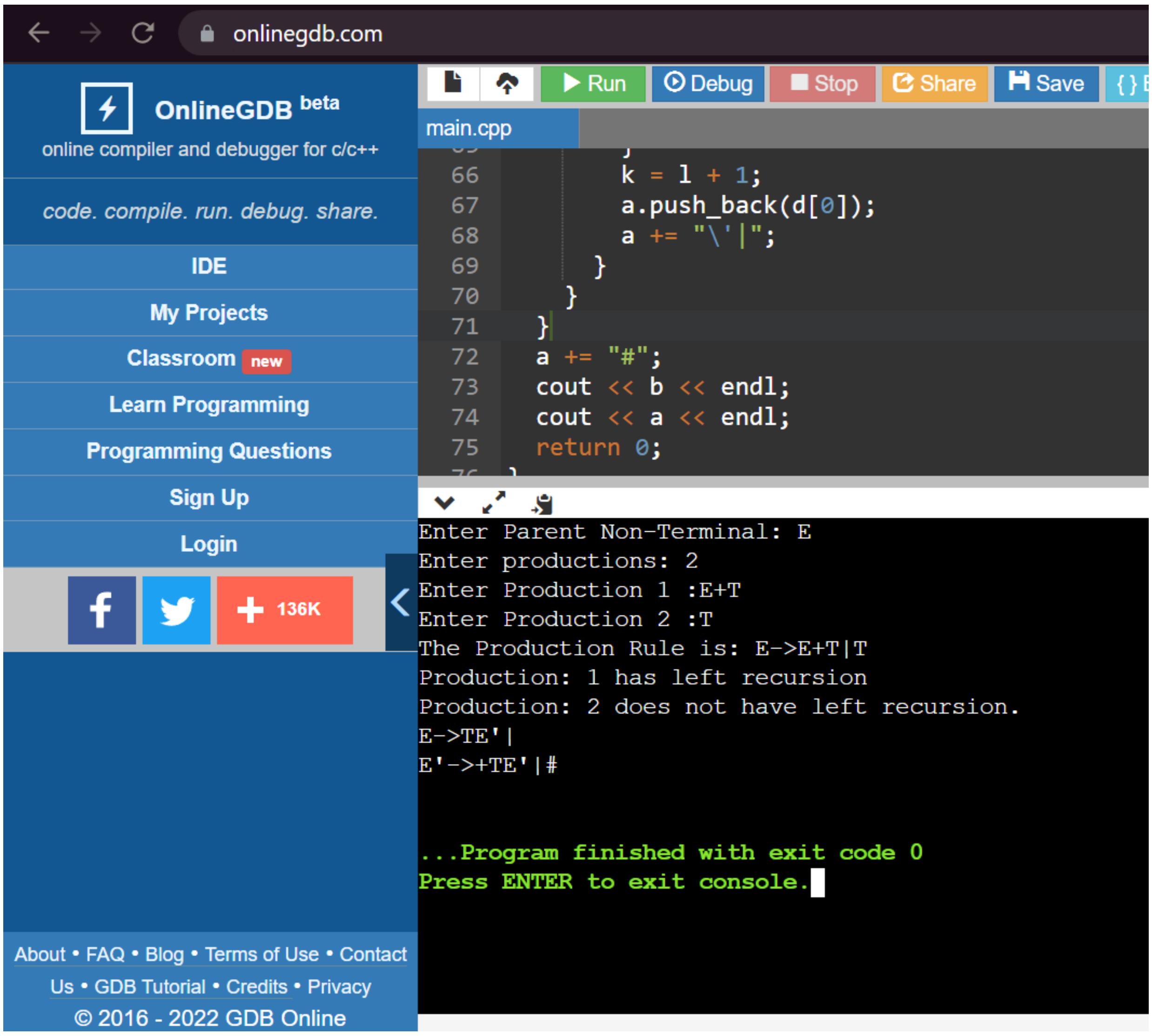
- Start the program
- Ask the user to enter the set of productions
- Check if the given grammar contains left recursion, if present then separate the production and start working on it.
  Let's take an example,
    - S-->Sa | Sb | c | d
- Introduce a new nonterminal and write it at the last of every terminal. We produce a new nonterminal S'and write new production as,
    - S-->cS' | dS'
- Write newly produced nonterminal in LHS and in RHS it can either produce or it can produce new production in which the terminals or non terminals which followed the previous LHS will be replaced by new nonterminal at last.
    - S'-->aS' | bS' | ε
- So after conversion the new equivalent production is,
    - S-->cS' | dS'
    - S'-->aS' | bS' | ε
- Stop

CODE:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
  int n, j, l, i, k;
  int length[10] = {};
  string d, a, b, flag;
  char c;
  cout<<"Enter Parent Non-Terminal: ";
  cin >> c;
  d.push_back(c);
  a += d + "\'->";
  d += "->";
  b += d;
  cout<<"Enter productions: ";
  cin >> n;
  for (int i = 0; i < n; i++)
  {
    cout<<"Enter Production ";
    cout<<i + 1<<" :";
    cin >> flag;
    length[i] = flag.size();
    d += flag;
    if (i != n - 1)
    {
```

```
    d += "|";
  }
}
cout<<"The Production Rule is: ";
cout<<d<<endl;
for (i = 0, k = 3; i < n; i++)
{
  if (d[0] != d[k])
  {
    cout<<"Production: "<< i + 1;
    cout<<" does not have left recursion.";
    cout<<endl;
    if (d[k] == '#')
    {
      b.push_back(d[0]);
      b += "\'";
    }
    else
    {
      for (j = k; j < k + length[i]; j++)
      {
        b.push_back(d[j]);
      }
      k = j + 1;
      b.push_back(d[0]);
      b += "\'|";
    }
    }
    else
    {
      cout<<"Production: "<< i + 1 ;
      cout<< " has left recursion";
      cout<< endl;
      if (d[k] != '#')
      {
        for (l = k + 1; l < k + length[i]; l++)
        {
          a.push_back(d[l]);
        }
        k = l + 1;
        a.push_back(d[0]);
        a += "\'|";
      }
    }
  }
  a += "#";
  cout << b << endl;
  cout << a << endl;
  return 0;
}
```

OUTPUT:



```cpp
                k = l + 1;
                a.push_back(d[0]);
                a += "\'|";
            }
        }
    }
    a += "#";
    cout << b << endl;
    cout << a << endl;
    return 0;
```

```
Enter Parent Non-Terminal: E
Enter productions: 2
Enter Production 1 :E+T
Enter Production 2 :T
The Production Rule is: E->E+T|T
Production: 1 has left recursion
Production: 2 does not have left recursion.
E->TE'|
E'->+TE'|#


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

The program for elimination of left recursion was successfully executed in C++.

IMPLEMENTATION OF LEFT FACTORING ELIMINATION

NAME: **RISHAL RAMESH** ***EXP NO:*** **4b**

REG NO: **RA1911030010084**

DATE: **17/02/2022**

AIM:

To study and implement left factoring elimination for the given productions.

LANGUAGE USED:

C++

ALGORITHM:

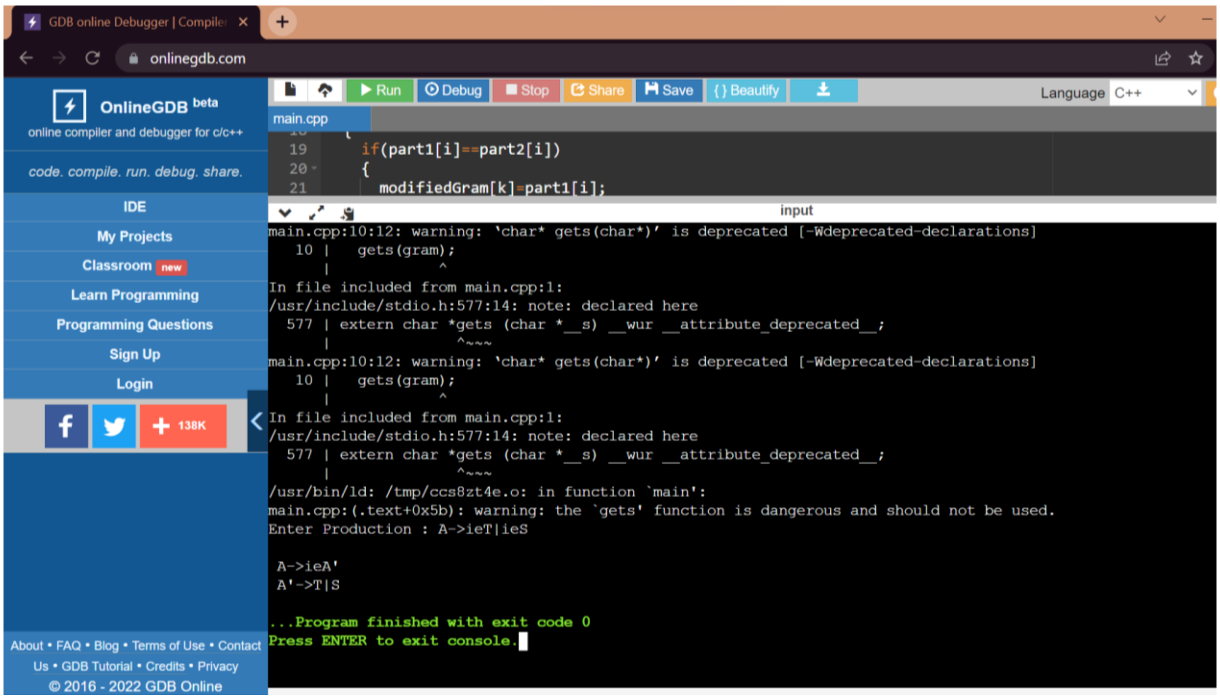
- Start
- Ask the user to enter the set of productions
- Check for common symbols in the given set of productions by comparing with:
    - A-> aB_1 | aB_2
- If found, replace the particular productions with:
    - A->aA'
    - A'->B_1 | B_2 | ε
- Display the output
- Stop

CODE:

```
#include<stdio.h>

#include<string.h>

#include <iostream>

using namespace std;
```

```
int main()
{
  char
gram[20],part1[20],part2[20],modifiedGram[20],newGram[20],tempGram[
20];
  int i,j=0,k=0,l=0,pos;
  cout<<"Enter Production : A->";
  gets(gram);
  for(i=0;gram[i]!='|';i++,j++)
    part1[j]=gram[i];
  part1[j]='\0';
  for(j=++i,i=0;gram[j]!='\0';j++,i++)
    part2[i]=gram[j];
  part2[i]='\0';
  for(i=0;i<strlen(part1)||i<strlen(part2);i++)
  {
    if(part1[i]==part2[i])
    {
      modifiedGram[k]=part1[i];
      k++;
      pos=i+1;
    }
  }
  for(i=pos,j=0;part1[i]!='\0';i++,j++){
    newGram[j]=part1[i];
  }
  newGram[j++]='|';
  for(i=pos;part2[i]!='\0';i++,j++){
    newGram[j]=part2[i];
  }
  modifiedGram[k]='A';
  modifiedGram[++k]='\0';
  newGram[j]='\0';
  cout<<"\n A->"<<modifiedGram<<"'";
  cout<<"\n A'->"<<newGram;
  return 0;
}
```

OUTPUT:



```
    if(part1[i]==part2[i])
    {
      modifiedGram[k]=part1[i];
```

```
main.cpp:10:12: warning: 'char* gets(char*)' is deprecated [-Wdeprecated-declarations]
   10 |     gets(gram);
      |            ^
In file included from main.cpp:1:
/usr/include/stdio.h:577:14: note: declared here
  577 | extern char *gets (char *__s) __wur __attribute_deprecated__;
      |              ^~~~
main.cpp:10:12: warning: 'char* gets(char*)' is deprecated [-Wdeprecated-declarations]
   10 |     gets(gram);
      |            ^
In file included from main.cpp:1:
/usr/include/stdio.h:577:14: note: declared here
  577 | extern char *gets (char *__s) __wur __attribute_deprecated__;
      |              ^~~~
/usr/bin/ld: /tmp/ccs8zt4e.o: in function `main':
main.cpp:(.text+0x5b): warning: the `gets' function is dangerous and should not be used.
Enter Production : A->ieT|ieS

 A->ieA'
 A'->T|S

...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

The program for elimination of left factoring was successfully executed in C++.