

## PREDICTIVE PARSING TABLE

**EX. NO. 6**

**Parth Langalia**

**Date:23/3/23**

**AIM:** To write a program to implement the predictive parsing table.

### ALGORITHM:

1. Start the program.
2. Initialize the required variables.
3. Get the number of coordinates and productions from the user.
4. Perform the following for (each production  $A \rightarrow \alpha$  in  $G$ ) { for (each terminal  $a$  in  $\text{FIRST}(\alpha)$ ) add  $A \rightarrow \alpha$  to  $M[A, a]$ ; if ( $\epsilon$  is in  $\text{FIRST}(\alpha)$ ) for (each symbol  $b$  in  $\text{FOLLOW}(A)$ ) add  $A \rightarrow \alpha$  to  $M[A, b]$ ;
5. Print the resulting stack.
6. Print if the grammar is accepted or not.
7. Exit the program.

## PROGRAM:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    char fin[10][20],st[10][20],ft[20][20],fol[20][20];
    int a=0,e,i,t,b,c,n,k,l=0,j,s,m,p;

    cout << ("enter the no. of nonterminals\n");
    scanf("%d",&n);
    cout << ("enter the productions in a grammar\n");
    for(i=0;i<n;i++)
        scanf("%s",st[i]);
    for(i=0;i<n;i++)
        fol[i][0]='\0';
    for(s=0;s<n;s++)
    {
        for(i=0;i<n;i++)
        {
            j=3;
            l=0;
            a=0;
l1:if(!((st[i][j]>64)&&(st[i][j]<91)))
            {
                for(m=0;m<l;m++)
                {
                    if(ft[i][m]==st[i][j])
                        goto s1;
                }
            }
        }
    }
}
```

```

    }
    ft[i][l]=st[i][j];
    l=l+1;
    s1:j=j+1;
}
else
{
    if(s>0)
    {
        while(st[i][j]!=st[a][0])
        {
            a++;
        }
        b=0;
        while(ft[a][b]!='\0')
        {
            for(m=0;m<l;m++)
            {
                if(ft[i][m]==ft[a][b])
                    goto s2;
            }
            ft[i][l]=ft[a][b];
            l=l+1;
            s2:b=b+1;
        }
    }
}
while(st[i][j]!='\0')
{
    if(st[i][j]=='|')

```

```

        {
            j=j+1;
            goto l1;
        }
        j=j+1;
    }

    ft[i][l]='\0';
}
}
cout << ("first \n");
for(i=0;i<n;i++)
    cout << ("FIRS[%c]=%s\n",st[i][0],ft[i]);
fol[0][0]='$';
for(i=0;i<n;i++)
{
    k=0;
    j=3;
    if(i==0)
        l=1;
    else
        l=0;
    k1:while((st[i][0]!=st[k][j])&&(k<n))
    {
        if(st[k][j]=='\0')
        {
            k++;
            j=2;
        }
        j++;
    }

```

```
}
```

```
j=j+1;
```

```
if(st[i][0]==st[k][j-1])
```

```
{
```

```
    if((st[k][j]!='|')&&(st[k][j]!='\0'))
```

```
    {
```

```
        a=0;
```

```
        if(!((st[k][j]>64)&&(st[k][j]<91)))
```

```
        {
```

```
            for(m=0;m<l;m++)
```

```
            {
```

```
                if(fol[i][m]==st[k][j])
```

```
                goto q3;
```

```
            }
```

```
            fol[i][l]=st[k][j];
```

```
            l++;
```

```
            q3::
```

```
        }
```

```
    else
```

```
    {
```

```
        while(st[k][j]!=st[a][0])
```

```
        {
```

```
            a++;
```

```
        }
```

```
        p=0;
```

```
        while(ft[a][p]!='\0')
```

```
        {
```

```
            if(ft[a][p]!='@')
```

```
            {
```

```

        for(m=0;m<l;m++)
        {
            if(fol[i][m]==ft[a][p])
                goto q2;
        }
        fol[i][l]=ft[a][p];
        l=l+1;
    }
    else
        e=1;
    q2:p++;
}
if(e==1)
{
    e=0;
    goto a1;
}
}
else
{
    a1:c=0;
    a=0;
    while(st[k][0]!=st[a][0])
    {
        a++;
    }
    while((fol[a][c]!='\0')&&(st[a][0]!=st[i][0]))
    {
        for(m=0;m<l;m++)

```

```

        {
            if(fol[i][m]==fol[a][c])
                goto q1;
        }
        fol[i][l]=fol[a][c];
        l++;
        q1:c++;
    }
}
goto k1;
}
fol[i][l]='\0';
}
cout << ("follow \n");
for(i=0;i<n;i++)
    cout << ("FOLLOW[%c]=%s\n",st[i][0],fol[i]);
cout << ("\n");
s=0;
for(i=0;i<n;i++)
{
    j=3;
    while(st[i][j]!='\0')
    {
        if((st[i][j-1]=='|')||(j==3))
        {
            for(p=0;p<=2;p++)
            {
                fin[s][p]=st[i][p];
            }
            t=j;

```

```

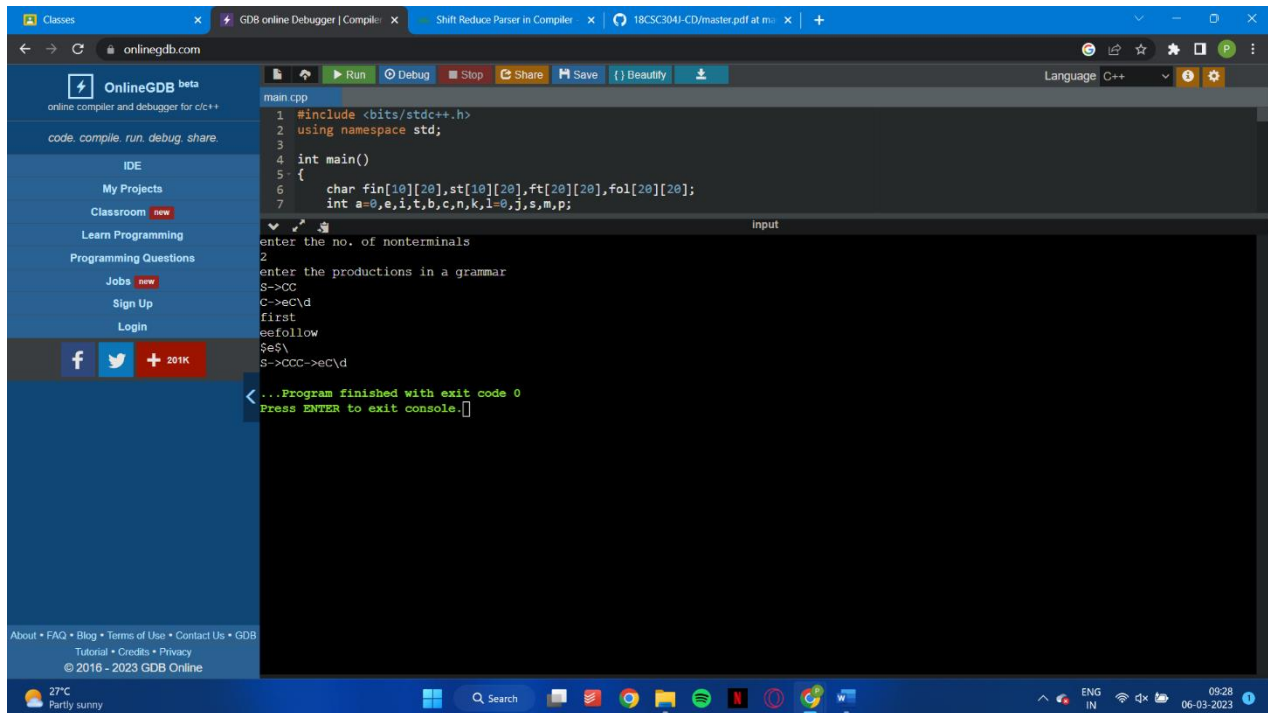
for(p=3;((st[i][j]!='')&&(st[i][j]!='\0'));p++)
{
    fin[s][p]=st[i][j];
    j++;
}
fin[s][p]='\0';
if(st[i][k]=='@')
{
    b=0;
    a=0;
    while(st[a][0]!=st[i][0])
    {
        a++;
    }
    while(fol[a][b]!='\0')
    {
        cout << ("M[%c,%c]=%s\n",st[i][0],fol[a][b],fin[s]);
        b++;
    }
}
else if(!((st[i][t]>64)&&(st[i][t]<91)))
    cout << ("M[%c,%c]=%s\n",st[i][0],st[i][t],fin[s]);
else
{
    b=0;
    a=0;
    while(st[a][0]!=st[i][3])
    {
        a++;
    }
}

```



```
        while(ft[a][b]!='\0')
        {
            cout << ("M[%c,%c]=%s\n",st[i][0],ft[a][b],fin[s]);
            b++;
        }
    }
    s++;
}
if(st[i][j]=='|')
j++;
}
}
```

## INPUT/OUTPUT :



The screenshot displays the OnlineGDB web interface. The browser tabs include 'Classes', 'GDB online Debugger | Compile', 'Shift Reduce Parser in Compiler', and '18CSC304I-CD/master.pdf at m...'. The URL bar shows 'onlinegdb.com'. The left sidebar contains navigation links: 'code, compile, run, debug, share', 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Jobs', 'Sign Up', and 'Login'. The main editor area shows a C++ program in 'main.cpp' with the following code:

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main()
5 {
6     char fin[10][20], st[10][20], ft[20][20], fol[20][20];
7     int a=0, e, i, t, b, c, n, k, l=0, j, s, m, p;
```

The input section shows the following interactions:

```
enter the no. of nonterminals
2
enter the productions in a grammar
S->CC
C->eC\d
first
eefollow
$e$
S->CCC->eC\d
```

The output section shows the following message:

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

The bottom status bar shows '27°C Partly sunny', 'ENG IN', and the date '06-03-2023'.

## RESULT :

The predictive parsing table has been implemented successfully.