Aim: Recursive Descent Parser

Input 1:

G->E
E->E+T|E-T|T
T->T*F|T/F|F
F->i|n

String: i-n*i

Output1:

```
D:\Semester7\CD_Lab\CD_Lab4_17CS01003>a.exe
Enter number of productions:4

Certain conditions for entering:
1.Nonterminals are single Capital alphabet and terminals are single lowercase alphabet
2.No space in defining production
3.The letter 'e' stands for epsilon.

Enter each production on a new line without space(E->E+T|T):
G->E
E->E+T|E-T|T
T->T*F|T/F|F
F->i|n
Enter the string to be parsed(enter string of terminals only):
i-n*i

Checking for immediate left recursion and updating productions...
Updated Productions:
G->E
E->TZ
T->FY
F->i|n
Z->+TZ|-TZ|e
Y->*FY|/FY|e

Left factoring the productions and updating...

Updated Productions:
G->E
E->TZ
T->FY
F->i|n
Z->+TZ|-TZ|e
Y->*FY|/FY|e
```

```
Z->+TZ|-TZ|e
Y->*FY|/FY|e

Checking if production E works
Encontered non terminal E
Checking if production TZ works
Encontered non terminal T
Checking if production FY works
Encontered non terminal F
Checking if production i works
Encontered terminal i
Encontered non terminal Y
Checking if production *FY works
Encontered terminal *
Encountered mismatch!!Backtracking
Checking if production /FY works
Encontered terminal /
Encountered mismatch!!Backtracking
Checking if production e works
Encontered epsilon
Encontered non terminal Z
Checking if production +TZ works
Encontered terminal +
Encountered mismatch!!Backtracking
Checking if production -TZ works
Encontered terminal -
Encontered non terminal T
Checking if production FY works
Encontered non terminal F
Checking if production i works
Encontered terminal i
Encountered mismatch!!Backtracking
Checking if production n works
Encontered terminal n
Encontered non terminal Y
Checking if production *FY works
Encontered terminal *
Encontered non terminal F
Checking if production i works
Encontered terminal i
Encontered non terminal Y
Checking if production *FY works
Encontered terminal *
Encountered mismatch!!Backtracking
```

```
Checking if production /FY works
Encontered terminal /
Encountered mismatch!!Backtracking
Checking if production e works
Encontered epsilon
Encontered non terminal Z
Checking if production +TZ works
Encontered terminal +
Encountered mismatch!!Backtracking
Checking if production -TZ works
Encontered terminal -
Encountered mismatch!!Backtracking
Checking if production e works
Encontered epsilon
There input code is correct!!
```

Input 2:

A->Aa|b

String: baa

Output2:

```
D:\Semester7\CD_Lab\CD_Lab4_17CS01003>a.exe
Enter number of productions:1

Certain conditions for entering:
1.Nonterminals are single Capital alphabet and terminals are single lowercase alphabet
2.No space in defining production
3.The letter 'e' stands for epsilon.

Enter each production on a new line without space(E->E+T|T):
A->Aa|b
Enter the string to be parsed(enter string of terminals only):
baa

Checking for immediate left recursion and updating productions...
Updated Productions:
A->bZ
Z->aZ|e

Left factoring the productions and updating...

Updated Productions:
A->bZ
Z->aZ|e

Checking if production bZ works
Encontered terminal b
Encontered non terminal Z
Checking if production aZ works
Encontered terminal a
Encontered non terminal Z
Checking if production aZ works
Encontered terminal a
Encontered non terminal Z
Checking if production aZ works
Encontered terminal a
Encountered mismatch!!Backtracking
Checking if production e works
Encontered epsilon
There input code is correct!!
```

Input3:

G->E
E->E+T|E-T|T
T->T*F|T/F|F
F->i|n

String: i**n

Output3:

```
D:\Semester7\CD_Lab\CD_Lab4_17CS01003>a.exe
Enter number of productions:4

Certain conditions for entering:
1.Nonterminals are single Capital alphabet and terminals are single lowercase alphabet
2.No space in defining production
3.The letter 'e' stands for epsilon.

Enter each production on a new line without space(E->E+T|T):
G->E
E->E+T|E-T|T
T->T*F|T/F|F
F->i|n
Enter the string to be parsed(enter string of terminals only):
i**n

Checking for immediate left recursion and updating productions...
Updated Productions:
G->E
E->TZ
T->FY
F->i|n
Z->+TZ|-TZ|e
Y->*FY|/FY|e

Left factoring the productions and updating...

Updated Productions:
G->E
E->TZ
T->FY
F->i|n
Z->+TZ|-TZ|e
Y->*FY|/FY|e
```

```
Checking if production E works
Encontered non terminal E
Checking if production TZ works
Encontered non terminal T
Checking if production FY works
Encontered non terminal F
Checking if production i works
Encontered terminal i
Encontered non terminal Y
Checking if production *FY works
Encontered terminal *
Encontered non terminal F
Checking if production i works
Encontered terminal i
Encountered mismatch!!Backtracking
Checking if production n works
Encontered terminal n
Encountered mismatch!!Backtracking
Encountered mismatch!!Backtracking
Checking if production /FY works
Encontered terminal /
Encountered mismatch!!Backtracking
Checking if production e works
Encontered epsilon
Encontered non terminal Z
Checking if production +TZ works
Encontered terminal +
Encountered mismatch!!Backtracking
Checking if production -TZ works
Encontered terminal -
Encountered mismatch!!Backtracking
Checking if production e works
Encontered epsilon
Sorry the input string has errors
```