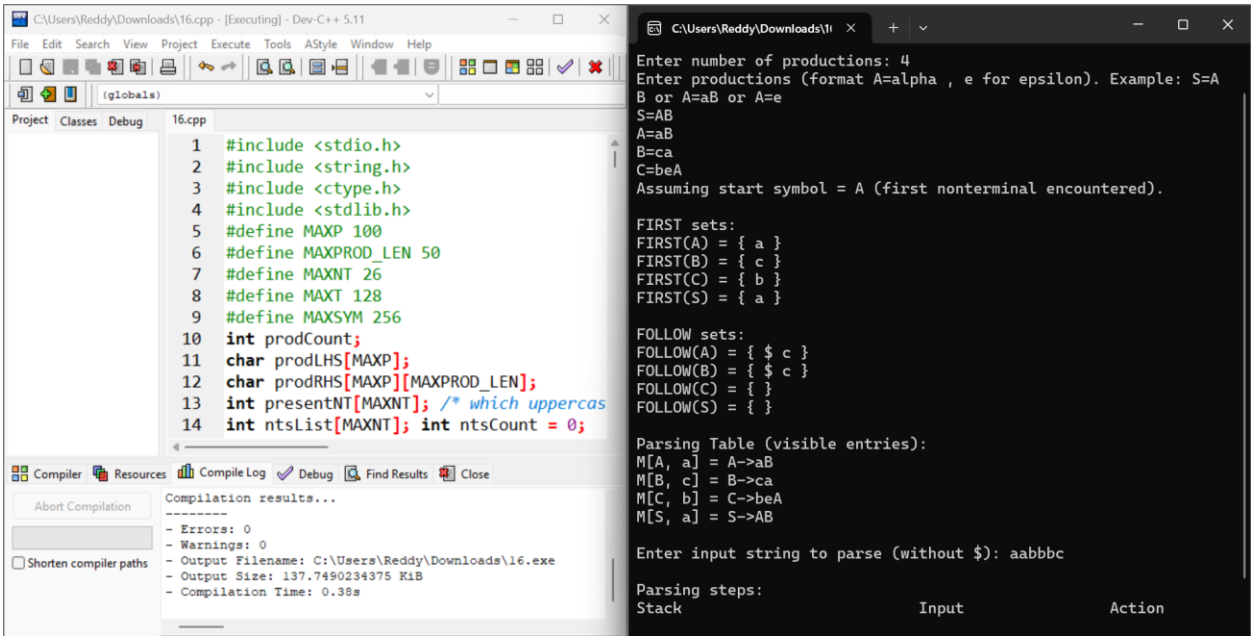
Exp 16: -Preadictive Parsing Steps

Aim: -To Implement a c programming language that calculates the FIRST and FOLLOW and predictive parsing for the given grammar.

Code:

```
#include <stdio.h>
#include <string.h>
char stack[50];
char input[50];
void push(char c) {
    int n=strlen(stack);
    stack[n]=c;
    stack[n+1]='\0';
}
void pop() {
    int n=strlen(stack);
    if(n>0) stack[n-1]='\0';
}
int main() {
    strcpy(stack, "E$")
    printf("Enter input (id+id): ");
    scanf("%s", input);
    strcat(input, "$");
    int i=0;
    while(1) {
        char X = stack[strlen(stack)-1];
        char a = input[i];
if(X=='$' && a=='$') {
            printf("Accepted\n");
            break;
```

```
        }
        if(X=='i' && a=='i') {
            pop();
            i++;
        }
        else if(X=='E') {
            pop();
            push('\''); push('E'); push('T');
        }
        else if(X=='\'') {
            pop();
            if(a=='+') { push('\''); push('E'); push('T'); push('+'); }
        }
        else if(X=='T') {
            pop();
            push('d'); push('i');
        }
        else {
            printf("Error\n");
            break;
        }
    }
    return 0;
}
```


C:\Users\Reddy\Downloads\16.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
16.cpp
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#define MAXP 100
#define MAXPROD_LEN 50
#define MAXNT 26
#define MAXT 128
#define MAXSYM 256
int prodCount;
char prodLHS[MAXP];
char prodRHS[MAXP][MAXPROD_LEN];
int presentNT[MAXNT]; /* which uppercas
int ntsList[MAXNT]; int ntsCount = 0;
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Reddy\Downloads\16.exe
- Output Size: 137.7490234375 KiB
- Compilation Time: 0.38s
C:\Users\Reddy\Downloads\1
Enter number of productions: 4
Enter productions (format A=alpha , e for epsilon). Example: S=A
B or A=aB or A=e
S=AB
A=aB
B=ca
C=beA
Assuming start symbol = A (first nonterminal encountered).
FIRST sets:
FIRST(A) = { a }
FIRST(B) = { c }
FIRST(C) = { b }
FIRST(S) = { a }
FOLLOW sets:
FOLLOW(A) = { $ c }
FOLLOW(B) = { $ c }
FOLLOW(C) = { }
FOLLOW(S) = { }
Parsing Table (visible entries):
M[A, a] = A->aB
M[B, c] = B->ca
M[C, b] = C->beA
M[S, a] = S->AB
Enter input string to parse (without $): aabbbc
Parsing steps:
Stack Input Action