**Write a C program for implementing the functionalities of predictive parser for the mini language**

**PRE-LAB QUESTIONS:**1. What is top-down parsing?  
2. What are the disadvantages of brute force method?  
3. What is context free grammar?  
4. What is parse tree?  
5. What is ambiguous grammar?  
6. What are the derivation methods to generate a string for the given grammar?  
7. What is the output of parse tree?

PROGRAM LOGIC:Read the input string.  
By using the FIRST AND FOLLOW values.  
Verify the FIRST of non-terminal and insert the production in the FIRST value  
If we have any @ terms in FIRST then insert the productions in FOLLOW values  
Constructing the predictive parser table

**INPUT & OUTPUT:**The following is the predictive parsing table for the following grammar:

S->A  
A->Bb  
A->Cd  
B->aB  
B->@  
C->Cc  
C->@

Predictive parsing table is  
------------------------------------------------------------------  
 a b c d $  
------------------------------------------------------------------  
S S->A S->A S->A S->A  
------------------------------------------------------------------  
A A->Bb A->Bb A->Cd A->Cd  
------------------------------------------------------------------  
B B->aB B->@ B->@ B->@  
------------------------------------------------------------------  
C C->@ C->@ C->@  
------------------------------------------------------------------

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 a | 1 b | 2 c | 3 d | 4 $ |
| 0 S | 0,0 | 0,1 | 0,2 | 0,3 | 0,4 |
| 1 A | 1,0 | 1,1 =A->Bb | 1,2 | 1,3 | 1,4 |
| 2 B | 2,0 | 2,1 | 2,2 | 2,3 | 2,4 |
| 3 C | 3,0 | 3,1 | 3,2 | 3,3 | 3,4 |

**LAB ASSIGNMENT:**

1. Write a program to compute FIRST for the following grammar?  
E🡪TE'  
E'🡪+TE'/î  
T🡪FT’  
T'🡪\*FT'/î  
F🡪(E)/i

2. Write a program to compute FIRST for the following grammar?  
S🡪iCtSS’  
S’🡪eS/ î

3. Write a program to construct predictive parsing table for the following grammar?  
S🡪iCtSS’  
S’🡪eS/ î

**POST LAB QUESTIONS**

1. What is Predictive parser?  
2. How many types of analysis can we do using Parser?  
3. What is Recursive Decent Parser?  
4. How many types of Parsers are there?  
5. What is LR Parser?

PROGRAM:#include<stdio.h>  
#include<string.h>  
char prol[7][10]={"S","A","A","B","B","C","C"};  
char pror[7][10]={"A","Bb","Cd","aB","@","Cc","@"};  
char prod[7][10]={"S->A","A->Bb","A->Cd","B->aB","B->@","C->Cc","C->@"};

char first[7][10]={"abcd","ab","cd","a@","@","c@","@"};  
char follow[7][10]={"$","$","$","a$","b$","c$","d$"};

char table[5][6][10];  
numr(char c)  
{  
switch(c)  
{  
case 'S': return 0;  
case 'A': return 1;  
case 'B': return 2;  
case 'C': return 3;  
case 'a': return 0;  
case 'b': return 1;  
case 'c': return 2;  
case 'd': return 3;  
case '$': return 4;  
}

return(2);  
}  
void main()  
{  
int i,j,k;  
  
for(i=0;i<5;i++)  
for(j=0;j<6;j++)  
strcpy(table[i][j]," ");  
printf("\nThe following is the predictive parsing table for the following grammar:\n");  
for(i=0;i<7;i++)  
printf("%s\n",prod[i]);  
printf("\nPredictive parsing table is\n");  
fflush(stdin);  
for(i=0;i<7;i++)  
{  
k=strlen(first[i]);  
for(j=0;j<10;j++)  
if(first[i][j]!='@')  
strcpy(table[numr(prol[i][0])+1][numr(first[i][j])+1],prod[i]);  
}  
for(i=0;i<7;i++)  
{  
if(strlen(pror[i])==1)  
{  
if(pror[i][0]=='@')  
{  
k=strlen(follow[i]);  
for(j=0;j<k;j++)  
strcpy(table[numr(prol[i][0])+1][numr(follow[i][j])+1],prod[i]);

}  
}  
}

strcpy(table[0][0]," ");  
strcpy(table[0][1],"a");  
strcpy(table[0][2],"b");  
strcpy(table[0][3],"c");  
strcpy(table[0][4],"d");  
strcpy(table[0][5],"$");  
strcpy(table[1][0],"S");  
strcpy(table[2][0],"A");  
strcpy(table[3][0],"B");  
strcpy(table[4][0],"C");  
printf("\n--------------------------------------------------------\n");  
for(i=0;i<5;i++)  
for(j=0;j<6;j++)  
{  
printf("%-10s",table[i][j]);  
if(j==5)  
printf("\n--------------------------------------------------------\n");  
}  
getch();  
}