Design LALR bottom up parser for a given language

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
[0-9]+ {yylval.dval=atof(yytext);
return DIGIT;
}
\n|. return yytext[0];
%%

%{
/*This YACC specification file generates the LALR parser for the program
considered in experiment 4.*/
#include<stdio.h>
%}
%union
{
double dval;
}
%token <dval> DIGIT
%type <dval> expr
%type <dval> term
%type <dval> factor
%%
line: expr '\n' {
printf("%g\n",$1);
}
;
expr: expr '+' term {$$=$1 + $3 ;}
| term
;
term: term '*' factor {$$=$1 * $3 ;}

| factor
;
factor: '(' expr ')' {$$=$2 ;}
| DIGIT
;
%%
int main(){
   yyparse();
}
```

```
        }
      }
    }
    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");
      }
}
```

Output:

```
The following grammar is used for Parsing Table:
S->A
A->Bb
A->Cd
B->aB
B->@
C->Cc
C->@

Predictive parsing table:

----------------------------------------------------------
          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->@
----------------------------------------------------------

Process returned 0 (0x0)   execution time : 0.049 s
Press any key to continue.
```