Experiment -20

Write a C program to compute TRAILING( ) – operator precedence parser for the given grammar

E → E + T | T

T → T * F | F

F → (E) | id

Program:

```
#include <stdio.h>
#include <string.h>
struct entry {
   char row;
   char col;
   char rel;};
struct entry table[50];
int top = -1;
char terminals[] = { '+', '*', '(', ')', 'i', '$' };
int tcount = 6;
void install(char row, char col, char rel)
{  top++;
   table[top].row = row;
   table[top].col = col;
   table[top].rel = rel;}
int main()
{
   int i, j;
   char prod[][4] = { "E", "E", "T", "T", "F", "F" };
   char rhs[][4]  = { "+T", "", "*F", "", "(E)", "i" };
   for(i = 0; i < 6; i++)
   {
      for(j = 0; j < strlen(rhs[i]); j++)
```

```
    {
        if(rhs[i][j] == '+' || rhs[i][j] == '*' ||
           rhs[i][j] == '(' || rhs[i][j] == ')' ||
           rhs[i][j] == 'i')
        {
            install(prod[i][0], rhs[i][j], 'T');}}}
  printf("\nOPERATOR PRECEDENCE TABLE:\n");
  for(i = 0; i <= top; i++)
  {
     printf("%c\t%c\t%c\n", table[i].row, table[i].col, table[i].rel);}
  printf("\nRelations:\n");
  char prev = ' ';
  for(i = 0; i <= top; i++)
  {
     if(table[i].row != prev)
     {
        prev = table[i].row;
        printf("\n%c -> ", prev);}
     printf("%c ", table[i].col);}
  printf("\n");
  return 0;}
```

Output: