**Experiment 10**

**Aim :-** Write a program to make RDP

**Input Program :-** Any Arithmetic operation

**Output :-** Valid parsed or not.

**Grammar:-**

E->TE'

E'->+TE'|e

T->FT'

T'->\*FT'|e

F->(E)|i

**C Code:-**

#include "stdio.h"

#include "string.h"

#include "stdlib.h"

#include "ctype.h"

char ip\_sym[15], ip\_ptr = 0, op[50], tmp[50];

void e\_prime();

void e();

void t\_prime();

void t();

void f();

void advance();

int n = 0;

void e()

{

strcpy(op, "TE'");

printf("E=%-25s", op);

printf("E->TE'\n");

t();

e\_prime();

}

void e\_prime()

{

int i, n = 0, l;

for (i = 0; i <= strlen(op); i++)

if (op[i] != 'e')

tmp[n++] = op[i];

strcpy(op, tmp);

l = strlen(op);

for (n = 0; n < l && op[n] != 'E'; n++)

;

if (ip\_sym[ip\_ptr] == '+')

{

i = n + 2;

do

{

op[i + 2] = op[i];

i++;

} while (i <= l);

op[n++] = '+';

op[n++] = 'T';

op[n++] = 'E';

op[n++] = 39;

printf("E=%-25s", op);

printf("E'->+TE'\n");

advance();

t();

e\_prime();

}

else

{

op[n] = 'e';

for (i = n + 1; i <= strlen(op); i++)

op[i] = op[i + 1];

printf("E=%-25s", op);

printf("E'->e");

}

}

void t()

{

int i, n = 0, l;

for (i = 0; i <= strlen(op); i++)

if (op[i] != 'e')

tmp[n++] = op[i];

strcpy(op, tmp);

l = strlen(op);

for (n = 0; n < l && op[n] != 'T'; n++)

;

i = n + 1;

do

{

op[i + 2] = op[i];

i++;

} while (i < l);

op[n++] = 'F';

op[n++] = 'T';

op[n++] = 39;

printf("E=%-25s", op);

printf("T->FT'\n");

f();

t\_prime();

}

void t\_prime()

{

int i, n = 0, l;

for (i = 0; i <= strlen(op); i++)

if (op[i] != 'e')

tmp[n++] = op[i];

strcpy(op, tmp);

l = strlen(op);

for (n = 0; n < l && op[n] != 'T'; n++)

;

if (ip\_sym[ip\_ptr] == '\*')

{

i = n + 2;

do

{

op[i + 2] = op[i];

i++;

} while (i < l);

op[n++] = '\*';

op[n++] = 'F';

op[n++] = 'T';

op[n++] = 39;

printf("E=%-25s", op);

printf("T'->\*FT'\n");

advance();

f();

t\_prime();

}

else

{

op[n] = 'e';

for (i = n + 1; i <= strlen(op); i++)

op[i] = op[i + 1];

printf("E=%-25s", op);

printf("T'->e\n");

}

}

void f()

{

int i, n = 0, l;

for (i = 0; i <= strlen(op); i++)

if (op[i] != 'e')

tmp[n++] = op[i];

strcpy(op, tmp);

l = strlen(op);

for (n = 0; n < l && op[n] != 'F'; n++)

;

if ((ip\_sym[ip\_ptr] == 'i') || (ip\_sym[ip\_ptr] == 'I'))

{

op[n] = 'i';

printf("E=%-25s", op);

printf("F->i\n");

advance();

}

else

{

if (ip\_sym[ip\_ptr] == '(')

{

advance();

e();

if (ip\_sym[ip\_ptr] == ')')

{

advance();

i = n + 2;

do

{

op[i + 2] = op[i];

i++;

} while (i <= l);

op[n++] = '(';

op[n++] = 'E';

op[n++] = ')';

printf("E=%-25s", op);

printf("F->(E)\n");

}

}

else

{

printf("\n\t syntax error");

exit(1);

}

}

}

void advance()

{

ip\_ptr++;

}

void main()

{

int i;

printf("\nGrammar without left recursion");

printf("\n\t\t E->TE' \n\t\t E'->+TE'|e \n\t\t T->FT' ");

printf("\n\t\t T'->\*FT'|e \n\t\t F->(E)|i");

printf("\n Enter the input expression:");

gets(ip\_sym);

printf("Expressions");

printf("\t Sequence of production rules\n");

e();

for (i = 0; i < strlen(ip\_sym); i++)

{

if (ip\_sym[i] != '+' && ip\_sym[i] != '\*' && ip\_sym[i] != '(' &&

ip\_sym[i] != ')' && ip\_sym[i] != 'i' && ip\_sym[i] != 'I')

{

printf("\nSyntax error");

break;

}

for (i = 0; i <= strlen(op); i++)

if (op[i] != 'e')

tmp[n++] = op[i];

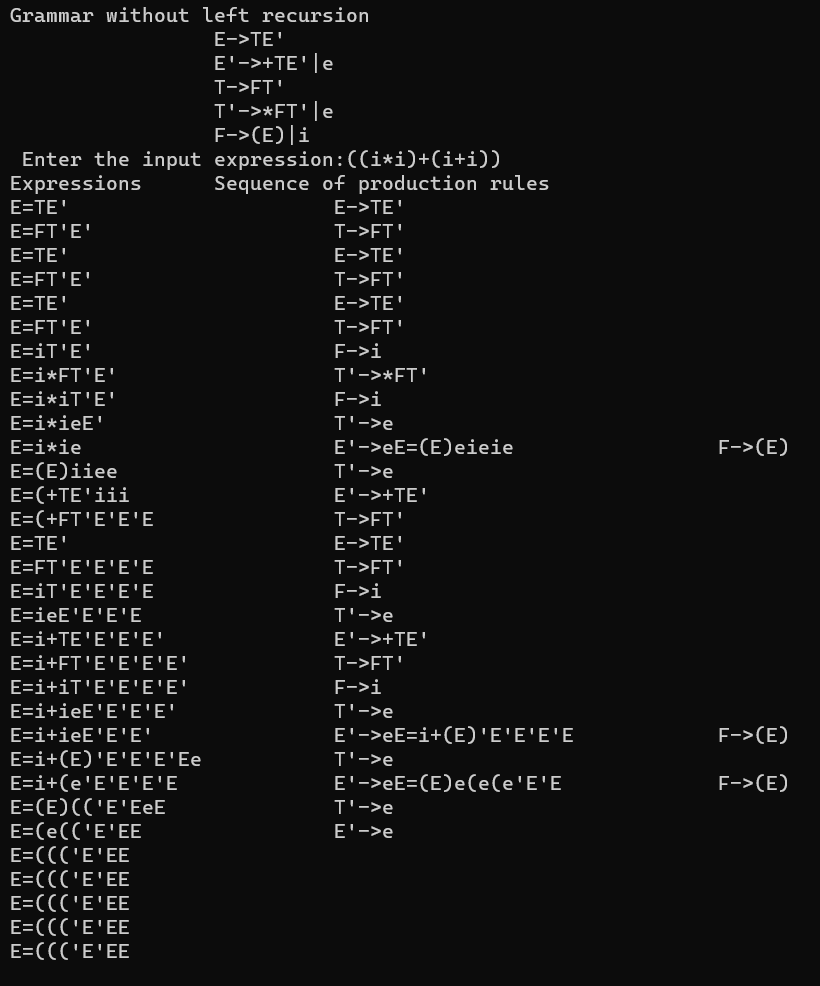
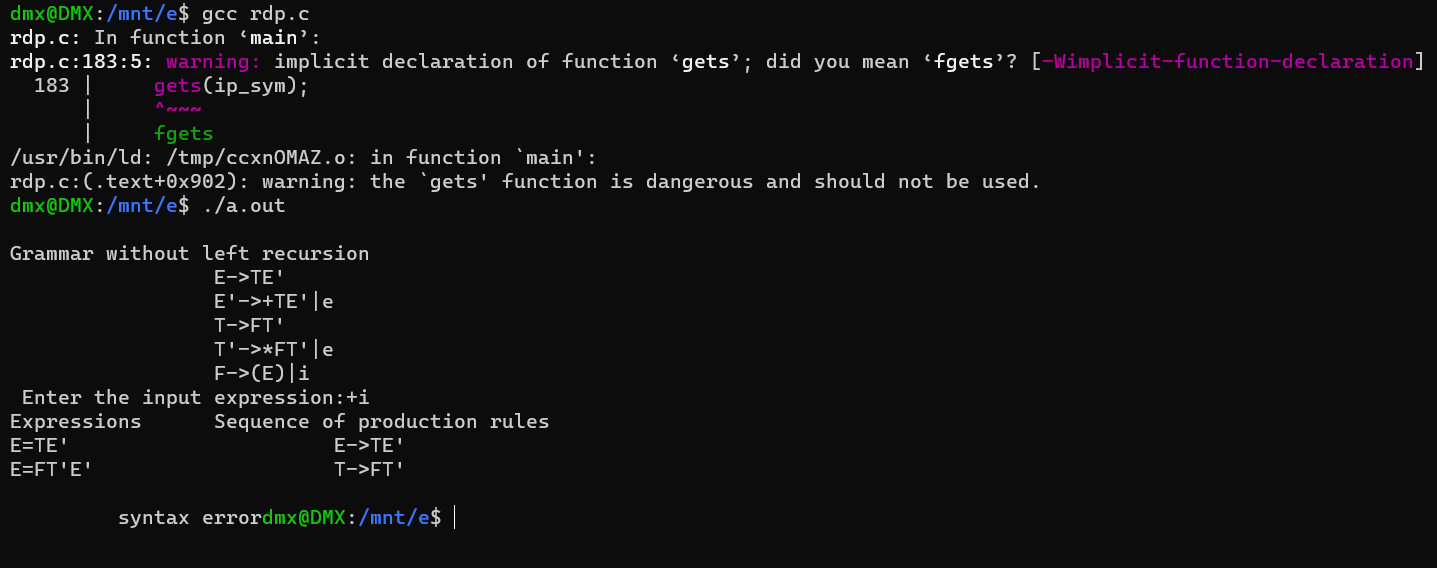
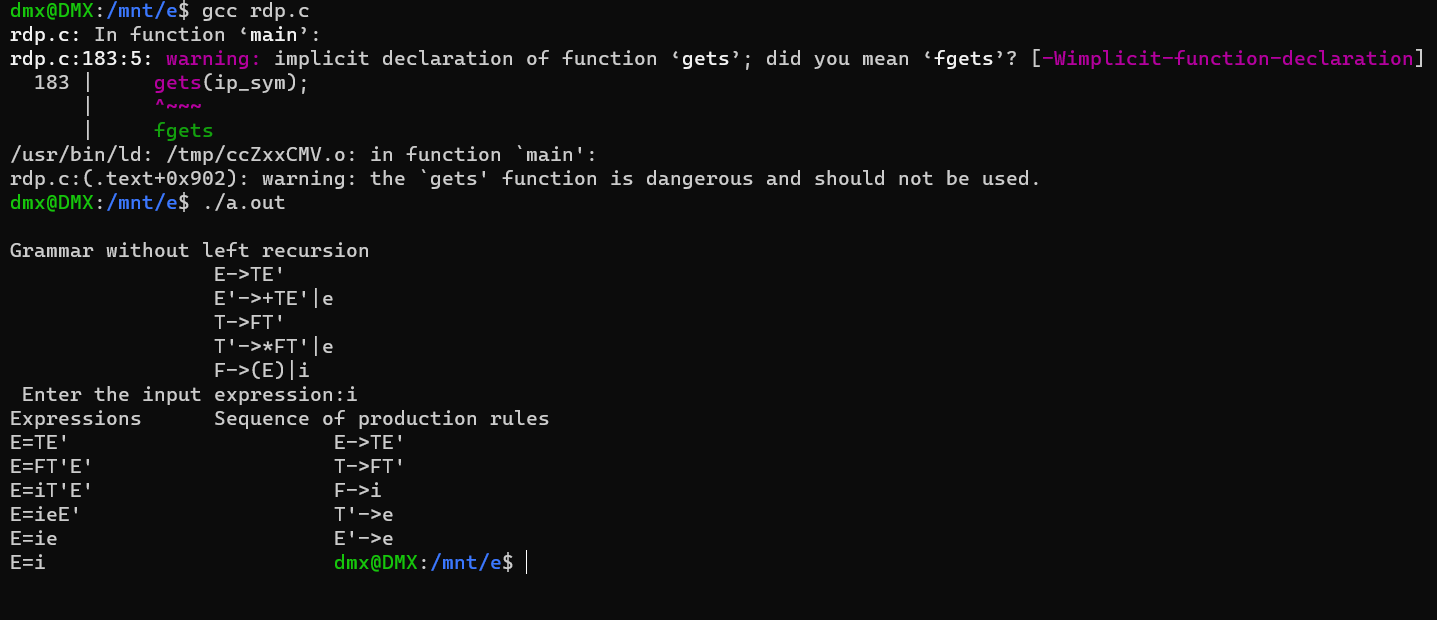
strcpy(op, tmp);

printf("\nE=%-25s", op);

}

}

**Output:-**

****