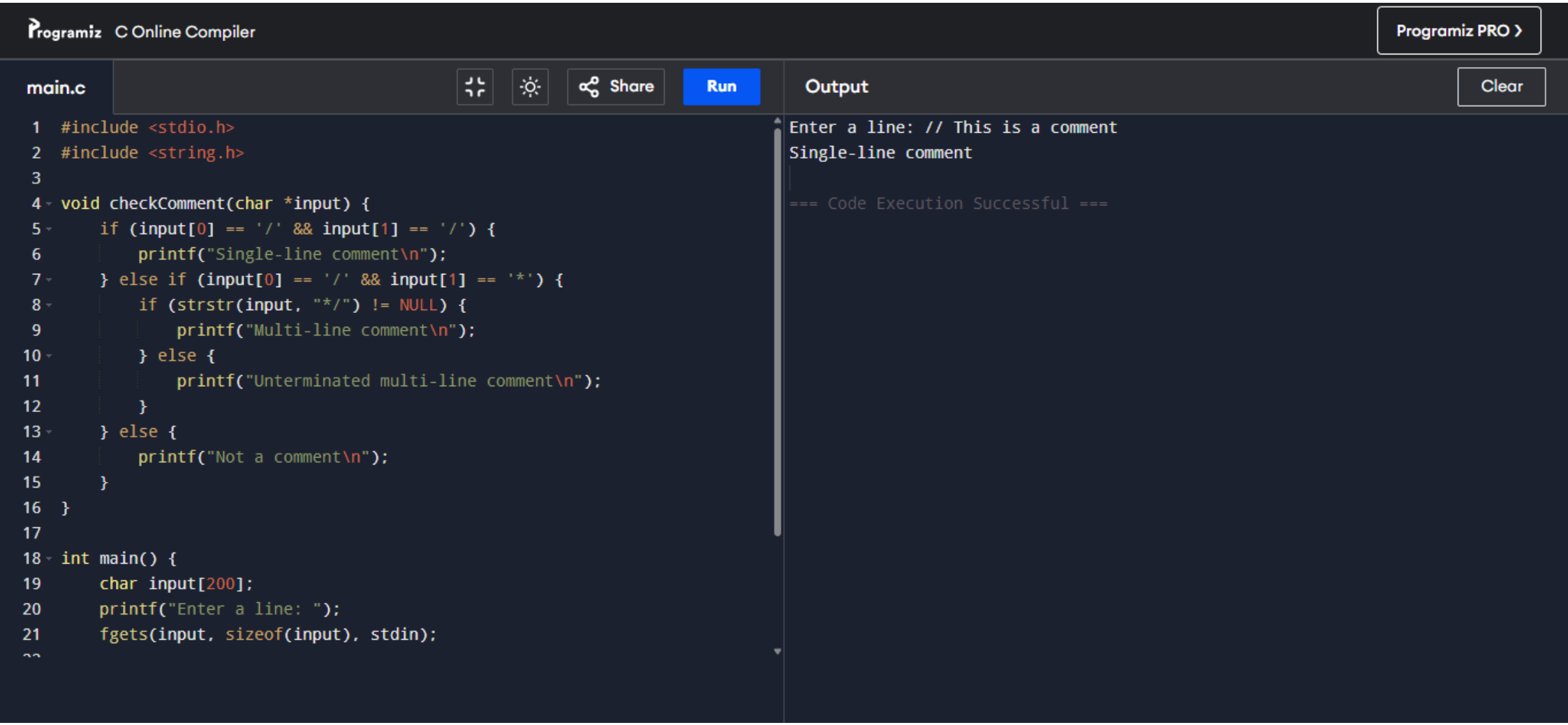
main.c

Share Run

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

void classifyToken(char *token, char identifiers[][20], int *idCount,
    char constants[][20], int *constCount, char operators[][5], int
    *opCount) {
    if (isdigit(token[0])) {
        strcpy(constants[*constCount], token);
        (*constCount)++;
    } else if (isalpha(token[0]) || token[0] == '_') {
        strcpy(identifiers[*idCount], token);
        (*idCount)++;
    } else {
        strcpy(operators[*opCount], token);
        (*opCount)++;
    }
}

int main() {
```

Output Clear

```
Enter the string: a = b + c * e + 100
Identifiers: a b c e
Constants: 100
Operators: = + * +

=== Code Execution Successful ===
```


Programiz C Online Compiler
Programiz PRO ›
main.c
Share
Run
Output
Clear
#include <stdio.h>
#include <string.h>
void checkComment(char *input) {
if (input[0] == '/' && input[1] == '/') {
printf("Single-line comment\n");
} else if (input[0] == '/' && input[1] == '*') {
if (strstr(input, "*/") != NULL) {
printf("Multi-line comment\n");
} else {
printf("Unterminated multi-line comment\n");
}
} else {
printf("Not a comment\n");
}
}
int main() {
char input[200];
printf("Enter a line: ");
fgets(input, sizeof(input), stdin);
Enter a line: // This is a comment
Single-line comment
=== Code Execution Successful ===

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

#define MAX_KEYWORDS 32

char keywords[MAX_KEYWORDS][10] = {
    "main", "auto", "break", "case", "char", "const", "continue",
        "default",
    "do", "double", "else", "enum", "extern", "float", "for", "goto",
    "if", "int", "long", "register", "return", "short", "signed",
    "sizeof", "static", "struct", "switch", "typedef",
    "unsigned", "void", "printf", "while"
};

char operators[] = "+-*/%=<>!&|";

int isKeyword(char *word) {
    for (int i = 0; i < MAX_KEYWORDS; i++) {
        if (strcmp(keywords[i], word) == 0)
```

Output

Clear

```
Enter a line: main int a b c = b + c printf % d
main is keyword
int is keyword
a is identifier
b is identifier
c is identifier
= is operator
b is identifier
+ is operator
c is identifier
printf is keyword
% is operator
d is identifier


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <string.h>

void checkOperator(char *op) {
    if (strcmp(op, "+") == 0) {
        printf("+ Addition Operator\n");
    } else if (strcmp(op, "-") == 0) {
        printf("- Subtraction Operator\n");
    } else if (strcmp(op, "*") == 0) {
        printf("* Multiplication Operator\n");
    } else if (strcmp(op, "/") == 0) {
        printf("/ Division Operator\n");
    } else if (strcmp(op, "=") == 0) {
        printf("= Assignment Operator\n");
    } else if (strcmp(op, "<") == 0) {
        printf("< Less than Operator\n");
    } else if (strcmp(op, ">") == 0) {
        printf("> Greater than Operator\n");
    } else if (strcmp(op, "<=") == 0) {
        printf("<= Less than or equal Operator\n");
    } else if (strcmp(op, ">=") == 0) {
```

Output

Clear

```
Enter any operator: <=
<= Less than or equal Operator


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void analyzeText(char *input) {
    int wordCount = 0, lineCount = 0, charCount = 0;
    int inWord = 0;

    for (int i = 0; input[i] != '\0'; i++) {
        charCount++;

        if (input[i] == '\n') {
            lineCount++;
        }

        if (isspace(input[i])) {
            inWord = 0;
        } else if (!inWord) {
            inWord = 1;
            wordCount++;
        }
```

Output

Clear

```
Enter text (Press Enter twice to stop):
void main()
{
    int a;
    int b;
    a = b + c;
    c = d * e;
}

Total number of words : 12
Total number of lines : 7
Total number of characters : 34


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO ›

main.c

Share

Run

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int isValidIdentifier(char *str) {
    if (!isalpha(str[0]) && str[0] != '_') {
        return 0;
    }
    for (int i = 1; str[i] != '\0'; i++) {
        if (!isalnum(str[i]) && str[i] != '_') {
            return 0;
        }
    }
    return 1;
}

int main() {
    char identifier[100];
    printf("Enter an identifier:");
    scanf("%s", identifier);

    if (isValidIdentifier(identifier)) {
```

Output

Clear

```
Enter an identifier:hrbv1215
Valid identifier


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include<stdio.h>
#include<ctype.h>
void FIRST(char[],char );
void addToResultSet(char[],char);
int numOfProductions;
char productionSet[10][10];
int main()
{
    int i;
    char choice;
    char c;
    char result[20];
    printf("How many number of productions ? :");
    scanf(" %d",&numOfProductions);
    for(i=0;i<numOfProductions;i++)//read production string eg: E=E
        +T
    {
        printf("Enter productions Number %d : ",i+1);
        scanf(" %s",productionSet[i]);
    }
    do
```

Output

Clear

```
How many number of productions ? :4
Enter productions Number 1 : S=AaAb
Enter productions Number 2 : S=BbBa
Enter productions Number 3 : A=$
Enter productions Number 4 : B=$
Find the FIRST of  :S
FIRST(S)= {  $  a  b }
press 'y' to continue : y
Find the FIRST of  :A
FIRST(A)= {  $ }
press 'y' to continue : y
Find the FIRST of  :B
FIRST(B)= {  $ }
press 'y' to continue : n


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include<stdio.h>
#include<ctype.h>
#include<string.h>
int limit, x = 0;
char production[10][10], array[10];

void find_first(char ch);
void find_follow(char ch);
void Array_Manipulation(char ch);

int main()
{
    int count;
    char option, ch;
    printf("\nEnter Total Number of Productions:\t");
    scanf("%d", &limit);
    for(count = 0; count < limit; count++)
    {
        printf("\nValue of Production Number [%d]:\t", count + 1
            );
        scanf("%s", production[count]);
```

Output

Clear

```
Enter Total Number of Productions:
4
Value of Production Number [1]: S=AaAb
Value of Production Number [2]: S=BbBa
Value of Production Number [3]: A=$
Value of Production Number [4]: B=$
Enter production Value to Find Follow:  S
Follow Value of S:      { $ }
To Continue, Press Y:   y
Enter production Value to Find Follow:  A
Follow Value of A:      { a b }
To Continue, Press Y:   y
Enter production Value to Find Follow:  B
Follow Value of B:      { b a }
To Continue, Press Y:   n


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share

Run

```c
#include<stdio.h>
#include<string.h>
#define SIZE 10
  int main () {
        char non_terminal;
        char beta,alpha;
        int num;
        char production[10][SIZE];
        int index=3; /* starting of the string following "->" */
        printf("Enter Number of Production : ");
        scanf("%d",&num);
        printf("Enter the grammar as E->E-A :\n");
        for(int i=0;i<num;i++){
            scanf("%s",production[i]);
        }
        for(int i=0;i<num;i++){
            printf("\nGRAMMAR : : : %s",production[i]);
            non_terminal=production[i][0];
            if(non_terminal==production[i][index]) {
                alpha=production[i][index+1];
                printf(" is left recursive.\n");
```

Output

Clear

```
Enter Number of Production : 2
Enter the grammar as E->E-A :
S->(L)|a
L->L,S|S

GRAMMAR : : : S->(L)|a is not left recursive.

GRAMMAR : : : L->L,S|S is left recursive.
Grammar without left recursion:
L->SL'
L'->,L'|E

=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include<stdio.h>
#include<string.h>

int main()
{
    char gram[20], part1[20], part2[20], modifiedGram[20],
        newGram[20];
    int i, j = 0, k = 0, l = 0, pos;

    // Input production
    printf("Enter Production: S->");
    gets(gram);

    // Extract part1 and part2
    for(i = 0; gram[i] != '|'; i++, j++)
        part1[j] = gram[i];
    part1[j] = '\0';

    for(j = ++i, i = 0; gram[j] != '\0'; j++, i++)
        part2[i] = gram[j];
    part2[i] = '\0';
```

Output

Clear

```
Enter Production : S->iEtS|iEtSeS|a

 S->iEtSX
 X->|eS|a

=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int cnt = 0;

struct symtab {
    char label[20];
    int addr;
} sy[50];

void insert();
int search(char *);
void display();
void modify();

int main() {
    int ch, val;
    char lab[20];

    do {
```

Output

Clear

```
1. Insert
2. Display
3. Search
4. Modify
5. Exit
Enter your choice: 1
Enter the label: a
Enter the address: 100

1. Insert
2. Display
3. Search
4. Modify
5. Exit
Enter your choice: 2

Symbol Table:
Label   Address
---------------
a   100
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share

Run

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

char input[100];
int pos = 0;

// Function declarations
void E();
void EPrime();
void T();
void TPrime();
void F();

void error() {
    printf("String is not accepted\n");
    exit(0);
}

// E -> T E'
void E() {
```

Output

Clear

```
Recursive descent parsing for the following grammar
E->TE'
E'->+TE'/@
T->FT'
T'->*FT'/@
F->(E)/ID
Enter the string to be checked: (a+b)*c
String is accepted
Enter the string to be checked: a/c+d
String is not accepted

=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share

Run

```c
#include <stdio.h>
#include <string.h>

char input[100]; // Stores the input string
int i = 0;       // Pointer to track input position

// Function to check if input matches the grammar S → aS | Sb | ab
int S() {
    if (input[i] == 'a') { // Matches 'a'
        i++; // Move to the next character
        if (S()) { // Recursive call for S → aS
            return 1;
        }
        if (input[i] == 'b') { // Matches 'Sb'
            i++; // Move to next character
            return 1;
        }
    } else if (input[i] == 'b') { // Base case: Matches "ab"
        i++; // Move past 'b'
        return 1;
    }
```

Output

Clear

```
The grammar is: S->aS, S->Sb, S->ab
Enter the string to be checked: aab
String accepted


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char ip_sym[15], stack[15];
int ip_ptr = 0, st_ptr = 0, len, i;
char temp[2], temp2[2];
char act[15];

void check();

int main() {
    printf("\n\t\t SHIFT REDUCE PARSER\n");

    printf("\n GRAMMAR\n");
    printf("\n E -> E+E\n E -> E/E\n E -> E*E\n E -> a/b\n");

    printf("\n Enter the input symbol: ");
    scanf("%s", ip_sym); // Using scanf instead of gets()

    printf("\n\tStack Implementation Table");
```

Output

Clear

```
        SHIFT REDUCE PARSER

GRAMMAR

E -> E+E
E -> E/E
E -> E*E
E -> a/b

Enter the input symbol: a+b

   Stack Implementation Table
Stack       Input Symbol        Action
-------------------------------------------------
$      a+b$            --
$a      +b$            shift a
$E      +b$            E->a
$E+          b$            shift+
$E+b          $            shiftb
$E+E          $            E->b
```

Programiz C Online Compiler

Programiz PRO >

main.c

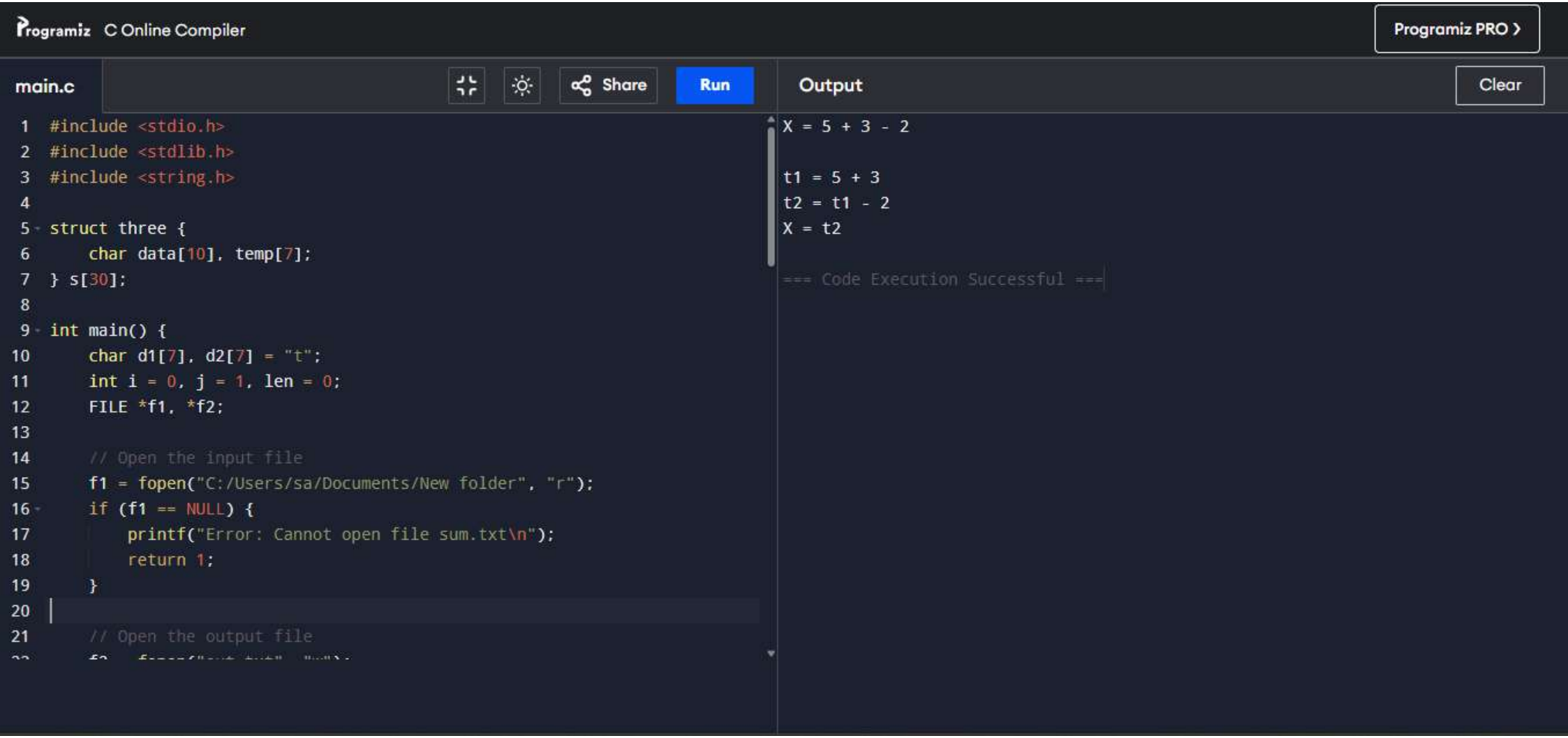
Share Run

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char input[50], stack[50];
int top = -1, i = 0;

char precedenceTable[9][9] = {
    /*  +    -    *    /    ^    i    (    )    $  */
    /* + */ '>', '>', '<', '<', '<', '<', '<', '>', '>',
    /* - */ '>', '>', '<', '<', '<', '<', '<', '>', '>',
    /* * */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
    /* / */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
    /* ^ */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
    /* i */ '>', '>', '>', '>', '>', 'e', 'e', '>', '>',
    /* ( */ '<', '<', '<', '<', '<', '<', '<', '>', 'e',
    /* ) */ '>', '>', '>', '>', '>', 'e', 'e', '>', '>',
    /* $ */ '<', '<', '<', '<', '<', '<', '<', '<', '>'
};

int getPrecedenceIndex(char symbol) {
```

Output

Clear

```
Enter the input string (append '$' at end): i*(i+i)*i$

STACK       INPUT         ACTION
-------------------------------------
$           i*(i+i)*i$    Shift
$i          *(i+i)*i$     Reduced: E->i
$E*         (i+i)*i$      Shift
$E*(        i+i)*i$       Shift
$E*(i       +i)*i$        Shift
$E*(E       +i)*i$        Reduced: E->i
$E*(E+      i)*i$         Shift
$E*(E+i     )*i$          Shift
$E*(E+E     )*i$          Reduced: E->i
$E*(E       )*i$          Reduced: E->E+E
$E*E        *i$           Reduced: E->(E)
$E*         i$            Shift
$E*i        $             Shift
$E*E        $             Reduced: E->i
$E          $             Reduced: E->E*E
$E$                       Accepted
```


Programiz C Online Compiler
Programiz PRO
main.c
Share
Run
Output
Clear
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct three {
char data[10], temp[7];
} s[30];
int main() {
char d1[7], d2[7] = "t";
int i = 0, j = 1, len = 0;
FILE *f1, *f2;
// Open the input file
f1 = fopen("C:/Users/sa/Documents/New folder", "r");
if (f1 == NULL) {
printf("Error: Cannot open file sum.txt\n");
return 1;
}
// Open the output file
X = 5 + 3 - 2
t1 = 5 + 3
t2 = t1 - 2
X = t2
=== Code Execution Successful ===

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

int main() {
    FILE *file;
    char filename[100], ch;
    int characters = 0, words = 0, lines = 0;
    int inWord = 0; // Flag to track if inside a word

    // Get filename from user
    printf("Enter the filename: ");
    scanf("%s", filename);

    // Open file
    file = fopen(filename, "r");
    if (file == NULL) {
        printf("ERROR: Cannot open file %s\n", filename);
        return 1;
    }
```

Output

Clear

```
Enter the filename: code.txt

Total number of characters: 51
Total number of words: 18
Total number of lines: 7


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share

Run

```c
#include <stdio.h>

int main() {
    int n, i;
    char a[50][10];

    printf("Enter the number of intermediate codes: ");
    scanf("%d", &n);
    getchar(); // Consume the newline character after scanf

    for (i = 0; i < n; i++) {
        printf("Enter the 3-address code %d: ", i + 1);
        fgets(a[i], sizeof(a[i]), stdin);
    }

    printf("\nThe generated code is:\n");

    for (i = 0; i < n; i++) {
        printf("mov %c, R%d\n", a[i][3], i);

        switch (a[i][4]) {
```

Output

Clear

```
enter the no: intermediate code:2
enter the 3 address code:1:a=b+c
enter the 3 address code:2:d=n*d
the generated code is:
mov a,R0
add c,R0
mov R0,a
mov n,R1
mul d,R1
mov R1,d


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <string.h>

char arr[18][3] = {
    {'E', '+', 'F'}, {'E', '*', 'F'}, {'E', '(', 'F'}, {'E', ')',
        'F'}, {'E', 'i', 'F'}, {'E', '$', 'F'},
    {'F', '+', 'F'}, {'F', '*', 'F'}, {'F', '(', 'F'}, {'F', ')',
        'F'}, {'F', 'i', 'F'}, {'F', '$', 'F'},
    {'T', '+', 'F'}, {'T', '*', 'F'}, {'T', '(', 'F'}, {'T', ')',
        'F'}, {'T', 'i', 'F'}, {'T', '$', 'F'}
};

char prod[] = "EETTFF";
char res[6][3] = {
    {'E', '+', 'T'}, {'T', '\0'}, {'T', '*', 'F'},
    {'F', '\0'}, {'(', 'E', ')'}, {'i', '\0'}
};

char stack[10][2]; // Increased stack size
int top = -1;
```

Output

Clear

```
Generated Table:
   E   +   T
   E   *   T
   E   (   T
   E   )   F
   E   i   T
   E   $   F
   F   +   F
   F   *   F
   F   (   T
   F   )   F
   F   i   T
   F   $   F
   T   +   F
   T   *   T
   T   (   T
   T   )   F
   T   i   T
   T   $   F
```

Programiz C Online Compiler

Programiz PRO ›

main.c

Share Run

```c
#include <stdio.h>
#include <string.h>

char arr[18][3] = {
    {'E', '+', 'F'}, {'E', '*', 'F'}, {'E', '(', 'F'}, {'E', ')',
        'F'}, {'E', 'i', 'F'}, {'E', '$', 'F'},
    {'F', '+', 'F'}, {'F', '*', 'F'}, {'F', '(', 'F'}, {'F', ')',
        'F'}, {'F', 'i', 'F'}, {'F', '$', 'F'},
    {'T', '+', 'F'}, {'T', '*', 'F'}, {'T', '(', 'F'}, {'T', ')',
        'F'}, {'T', 'i', 'F'}, {'T', '$', 'F'}
};

char prod[6] = "EETTFF";
char res[6][3] = {
    {'E', '+', 'T'}, {'T', '\0', '\0'}, {'T', '*', 'F'},
    {'F', '\0', '\0'}, {'(', 'E', ')'}, {'i', '\0', '\0'}
};

char stack[10][2]; // Stack size increased
int top = -1;
```

Output

Clear

```
Generated Table:
   E   +   T
   E   *   T
   E   (   F
   E   )   T
   E   i   T
   E   $   F
   F   +   F
   F   *   F
   F   (   F
   F   )   T
   F   i   T
   F   $   F
   T   +   F
   T   *   T
   T   (   F
   T   )   T
   T   i   T
   T   $   F
```