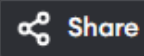


main.c



Share

Run

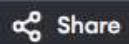
Output

```
1 #include<stdio.h>
2 #include<ctype.h>
3 #include<string.h>
4 int main()
5 {
6     int i,ic=0,m,cc=0,oc=0,j;
7     char b[30],operators[30],identifiers[30],constants[30];
8     printf("enter the string : ");
9     scanf("%[^\n]s",&b);
10    for(i=0;i<strlen(b);i++)
11    {
12        if(isspace(b[i]))
13        {
14            continue;
15        }
16        else if(isalpha(b[i]))
17        {
18            identifiers[ic] =b[i];
19            ic++;
20        }
21        else if(isdigit(b[i]))
22        {
```

```
enter the string : Enter the string: a = b - c * e + 89
Identifiers: a b c e
Constants: 89
Operators: = - * +
```

=== Code Execution Successful ===

main.c



Share

Run

Output

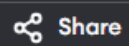
Clear

```
1 #include <stdio.h>
2 #include <string.h>
3
4 void checkComment(char *input) {
5     if (input[0] == '/' && input[1] == '/') {
6         printf("Single-line comment\n");
7     } else if (input[0] == '/' && input[1] == '*') {
8         if (strstr(input, "*/") != NULL) {
9             printf("Multi-line comment\n");
10        } else {
11            printf("Unterminated multi-line comment\n");
12        }
13    } else {
14        printf("Not a comment\n");
15    }
16 }
17
18 int main() {
19     char input[200];
20     printf("Enter a line: ");
21     fgets(input, sizeof(input), stdin);
22 }
```

Enter a line: /* This is a multi-line comment */
Multi-line comment

=== Code Execution Successful ===

main.c



Share

Run

Output

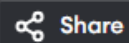
Clear

```
1 #include <stdio.h>
2 #include <ctype.h>
3 #include <string.h>
4 #include <stdlib.h>
5
6 #define MAX_KEYWORDS 32
7
8 char keywords[MAX_KEYWORDS][10] = {
9     "main", "auto", "break", "case", "char", "const", "continue",
10    "default",
11    "do", "double", "else", "enum", "extern", "float", "for", "goto",
12    "if", "int", "long", "register", "return", "short", "signed",
13    "sizeof", "static", "struct", "switch", "typedef",
14    "unsigned", "void", "printf", "while"
15 };
16
17 char operators[] = "+-*/%=<>!&|";
18
19 int isKeyword(char *word) {
20     for (int i = 0; i < MAX_KEYWORDS; i++) {
21         if (strcmp(keywords[i], word) == 0)
```

```
Enter a line: float x = 10 + y * z;
float is keyword
x is identifier
= is operator
10 is identifier
+ is operator
y is identifier
* is operator
z is identifier
```

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

main.c



Run

Output

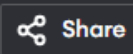
Clear

```
1 #include <stdio.h>
2 #include <string.h>
3
4 void checkOperator(char *op) {
5     if (strcmp(op, "+") == 0) {
6         printf("+ Addition Operator\n");
7     } else if (strcmp(op, "-") == 0) {
8         printf("- Subtraction Operator\n");
9     } else if (strcmp(op, "*") == 0) {
10        printf("* Multiplication Operator\n");
11    } else if (strcmp(op, "/") == 0) {
12        printf("/ Division Operator\n");
13    } else if (strcmp(op, "=") == 0) {
14        printf("=" Assignment Operator\n");
15    } else if (strcmp(op, "<") == 0) {
16        printf("< Less than Operator\n");
17    } else if (strcmp(op, ">") == 0) {
18        printf("> Greater than Operator\n");
19    } else if (strcmp(op, "<=") == 0) {
20        printf("<= Less than or equal Operator\n");
21    } else if (strcmp(op, ">=") == 0) {
22        printf(">= Greater than or equal Operator\n");
23    }
```

Enter any operator: +
+ Addition Operator

=== Code Execution Successful ===

main.c



Run

Output

Clear

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <ctype.h>
4
5 void analyzeText(char *input) {
6     int wordCount = 0, lineCount = 0, charCount = 0;
7     int inWord = 0;
8
9     for (int i = 0; input[i] != '\0'; i++) {
10         charCount++;
11
12         if (input[i] == '\n') {
13             lineCount++;
14         }
15
16         if (isspace(input[i])) {
17             inWord = 0;
18         } else if (!inWord) {
19             inWord = 1;
20             wordCount++;
21         }
22     }
23 }
```

Enter text (Press Enter twice to stop):

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

```
int main()
```

```
{
```

```
    printf("Hello, World!");
```

```
    return 0;
```

```
}
```

Total number of words : 10

Total number of lines : 6

Total number of characters : 45

=== Code Execution Successful ===

main.c



Share

Run

Output

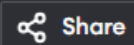
Clear

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <ctype.h>
4
5 int isValidIdentifier(char *str) {
6     if (!isalpha(str[0]) && str[0] != '_') {
7         return 0;
8     }
9     for (int i = 1; str[i] != '\0'; i++) {
10         if (!isalnum(str[i]) && str[i] != '_') {
11             return 0;
12         }
13     }
14     return 1;
15 }
16
17 int main() {
18     char identifier[100];
19     printf("Enter an identifier:");
20     scanf("%s", identifier);
21
22     if (isValidIdentifier(identifier)) {
23         printf("Valid identifier\n");
24     } else {
25         printf("Invalid identifier\n");
26     }
27 }
```

Enter an identifier:1654bvhugvyu
Invalid identifier

=== Code Execution Successful ===

main.c



Run

Output

Clear

```
1 #include<stdio.h>
2 #include<ctype.h>
3 void FIRST(char[],char );
4 void addToResultSet(char[],char);
5 int numOfProductions;
6 char productionSet[10][10];
7 int main()
8 {
9     int i;
10    char choice;
11    char c;
12    char result[20];
13    printf("How many number of productions ? :");
14    scanf(" %d",&numOfProductions);
15    for(i=0;i<numOfProductions;i++)//read production string eg: E=E
        +T
16    {
17        printf("Enter productions Number %d : ",i+1);
18        scanf(" %s",productionSet[i]);
19    }
20    do
21    {
```

```
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 ===
```

main.c

Share

Run

```
1  #include<stdio.h>
2  #include<ctype.h>
3  #include<string.h>
4  int limit, x = 0;
5  char production[10][10], array[10];
6
7  void find_first(char ch);
8  void find_follow(char ch);
9  void Array_Manipulation(char ch);
10
11 int main()
12 {
13     int count;
14     char option, ch;
15     printf("\nEnter Total Number of Productions:\t");
16     scanf("%d", &limit);
17     for(count = 0; count < limit; count++)
18     {
19         printf("\nValue of Production Number [%d]:\t", count + 1
20             );
21         scanf("%s", production[count]);
```

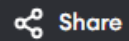
Output

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 ===

main.c



Share

Run

Output

Clear

```
1  #include<stdio.h>
2  #include<string.h>
3  #define SIZE 10
4  int main () {
5      char non_terminal;
6      char beta,alpha;
7      int num;
8      char production[10][SIZE];
9      int index=3; /* starting of the string following "->" */
10     printf("Enter Number of Production : ");
11     scanf("%d",&num);
12     printf("Enter the grammar as E->E-A :\n");
13     for(int i=0;i<num;i++){
14         scanf("%s",production[i]);
15     }
16     for(int i=0;i<num;i++){
17         printf("\nGRAMMAR : : : %s",production[i]);
18         non_terminal=production[i][0];
19         if(non_terminal==production[i][index]) {
20             alpha=production[i][index+1];
21             printf(" is left recursive.\n");
22         }
23     }
```

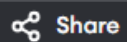
```
Enter Number of Production : 2
Enter the grammar as E->E-A :
S->(L)|a
L->L,S|S

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 ===
```

main.c



Share

Run

Output

Clear

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char gram[20], part1[20], part2[20], modifiedGram[20],
        newGram[20];
7      int i, j = 0, k = 0, l = 0, pos;
8
9      // Input production
10     printf("Enter Production: S->");
11     gets(gram);
12
13     // Extract part1 and part2
14     for(i = 0; gram[i] != '|'; i++, j++)
15         part1[j] = gram[i];
16     part1[j] = '\0';
17
18     for(j = ++i, i = 0; gram[j] != '\0'; j++, i++)
19         part2[i] = gram[j];
20     part2[i] = '\0';
21
```

Enter Production : S->iEtS|iEtSeS|a

S->iEtSX

X->|eS|a|

=== Code Execution Successful ===

main.c

Share

Run

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int cnt = 0;
6
7  struct symtab {
8      char label[20];
9      int addr;
10 } sy[50];
11
12 void insert();
13 int search(char *);
14 void display();
15 void modify();
16
17 int main() {
18     int ch, val;
19     char lab[20];
20
21     do {
22         printf("\n1. Insert\n2. Display\n3. Search\n4. Modify\n5. Exit\n");
                Enter your choice: 1
                Enter the label: x
                Enter the address: 300

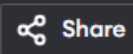
                1. Insert
                2. Display
                3. Search
                4. Modify
                5. Exit
                Enter your choice: 2

                Symbol Table:
                Label    Address
                -----
                x        300
            } while (ch != 5);
        }
```

Output

Clear

main.c



Run

Output

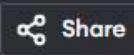
Clear

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <ctype.h>
4
5 char input[100];
6 int pos = 0;
7
8 // Function declarations
9 void E();
10 void EPrime();
11 void T();
12 void TPrime();
13 void F();
14
15 void error() {
16     printf("String is not accepted\n");
17     exit(0);
18 }
19
20 // E -> T E'
21 void E() {
22     T();
```

```
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 ===
```

main.c



Run

Output

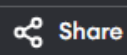
Clear

```
1 #include <stdio.h>
2 #include <string.h>
3
4 char input[100]; // Stores the input string
5 int i = 0;       // Pointer to track input position
6
7 // Function to check if input matches the grammar S → aS | Sb | ab
8 int S() {
9     if (input[i] == 'a') { // Matches 'a'
10         i++; // Move to the next character
11         if (S()) { // Recursive call for S → aS
12             return 1;
13         }
14     }
15     if (input[i] == 'b') { // Matches 'Sb'
16         i++; // Move to next character
17         return 1;
18     }
19     } else if (input[i] == 'b') { // Base case: Matches "ab"
20         i++; // Move past 'b'
21         return 1;
22     }
23     return 0; // If none of the above match, return 0 (failure)
24 }
```

The grammar is: $S \rightarrow aS$, $S \rightarrow Sb$, $S \rightarrow ab$
Enter the string to be checked: abb
String not accepted

=== Code Execution Successful ===

main.c



Run

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4
5 char ip_sym[15], stack[15];
6 int ip_ptr = 0, st_ptr = 0, len, i;
7 char temp[2], temp2[2];
8 char act[15];
9
10 void check();
11
12 int main() {
13     printf("\n\t\t SHIFT REDUCE PARSER\n");
14
15     printf("\n GRAMMAR\n");
16     printf("\n E -> E+E\n E -> E/E\n E -> E*E\n E -> a/b\n");
17
18     printf("\n Enter the input symbol: ");
19     scanf("%s", ip_sym); // Using scanf instead of gets()
20
21     printf("\n\tStack Implementation Table");
22     printf("\n\tStack\tInput Symbol\tAction\n");
```

Output

Clear

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
\$E+	\$	shift+
\$E+	\$	REJECT

=== Code Execution Successful ===

main.c



Share

Run

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4
5 char input[50], stack[50];
6 int top = -1, i = 0;
7
8 char precedenceTable[9][9] = {
9     /* + - * / ^ i ( ) $ */
10    /* + */ '>', '>', '<', '<', '<', '<', '<', '>', '>',
11    /* - */ '>', '>', '<', '<', '<', '<', '<', '>', '>',
12    /* * */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
13    /* / */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
14    /* ^ */ '>', '>', '>', '>', '<', '<', '<', '>', '>',
15    /* i */ '>', '>', '>', '>', '>', '>', 'e', 'e', '>', '>',
16    /* ( */ '<', '<', '<', '<', '<', '<', '<', '>', 'e',
17    /* ) */ '>', '>', '>', '>', '>', '>', 'e', 'e', '>', '>',
18    /* $ */ '<', '<', '<', '<', '<', '<', '<', '<', '<', '>'
19 };
20
21 int getPrecedenceIndex(char symbol) {
22     int index = -1;
23     for (int i = 0; i < 9; i++) {
24         if (symbol == precedenceTable[i][0]) {
25             index = i;
26         }
27     }
28     return index;
29 }
```

Output

Clear

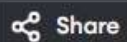
Enter the input string (append '\$' at end): i+i*i\$

STACK	INPUT	ACTION

\$	i+i*i\$	Shift
\$i	+i*i\$	Reduced: E->i
\$E+	i*i\$	Shift
\$E+i	*i\$	Shift
\$E+E	*i\$	Reduced: E->i
\$E+E*	i\$	Shift
\$E+E*i	\$	Shift
\$E+E*E	\$	Reduced: E->i
\$E+E	\$	Reduced: E->E*E
\$E	\$	Reduced: E->E+E
\$E\$		Accepted

=== Code Execution Successful ===

main.c



Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 struct three {
6     char data[10], temp[7];
7 } s[30];
8
9 int main() {
10     char d1[7], d2[7] = "t";
11     int i = 0, j = 1, len = 0;
12     FILE *f1, *f2;
13
14     // Open the input file
15     f1 = fopen("C:/Users/sa/Documents/New folder", "r");
16     if (f1 == NULL) {
17         printf("Error: Cannot open file sum.txt\n");
18         return 1;
19     }
20
21     // Open the output file
22     f2 = fopen("C:/Users/sa/Documents/New folder", "w");
```

```
Y = 10 - 4 + 2
t1 = 10 - 4
t2 = t1 + 2
Y = t2
```

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


main.c



Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ctype.h>
4
5 int main() {
6     FILE *file;
7     char filename[100], ch;
8     int characters = 0, words = 0, lines = 0;
9     int inWord = 0; // Flag to track if inside a word
10
11     // Get filename from user
12     printf("Enter the filename: ");
13     scanf("%s", filename);
14
15     // Open file
16     file = fopen(filename, "r");
17     if (file == NULL) {
18         printf("ERROR: Cannot open file %s\n", filename);
19         return 1;
20     }
21
22     // Read file and count characters, words, and lines
```

Enter the filename: code.txt

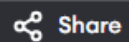
Total number of characters: 51

Total number of words: 18

Total number of lines: 7

=== Code Execution Successful ===

main.c



Share

Run

Output

Clear

```
1  #include <stdio.h>
2
3  int main() {
4      int n, i;
5      char a[50][10];
6
7      printf("Enter the number of intermediate codes: ");
8      scanf("%d", &n);
9      getchar(); // Consume the newline character after scanf
10
11     for (i = 0; i < n; i++) {
12         printf("Enter the 3-address code %d: ", i + 1);
13         fgets(a[i], sizeof(a[i]), stdin);
14     }
15
16     printf("\nThe generated code is:\n");
17
18     for (i = 0; i < n; i++) {
19         printf("mov %c, R%d\n", a[i][3], i);
20
21         switch (a[i][4]) {
22             // ...
23         }
24     }
25 }
```

```
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 ===
```

main.c

Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <string.h>
3
4 char arr[18][3] = {
5     {'E', '+', 'F'}, {'E', '*', 'F'}, {'E', '(', 'F'}, {'E', ')',
6     'F'}, {'E', 'i', 'F'}, {'E', '$', 'F'},
7     {'F', '+', 'F'}, {'F', '*', 'F'}, {'F', '(', 'F'}, {'F', ')',
8     'F'}, {'F', 'i', 'F'}, {'F', '$', 'F'},
9     {'T', '+', 'F'}, {'T', '*', 'F'}, {'T', '(', 'F'}, {'T', ')',
10    'F'}, {'T', 'i', 'F'}, {'T', '$', 'F'}
11 };
12
13 char prod[] = "EETFFF";
14 char res[6][3] = {
15     {'E', '+', 'T'}, {'T', '\0'}, {'T', '*', 'F'},
16     {'F', '\0'}, {'(', 'E', ')'}, {'i', '\0'}
17 };
18
19 char stack[10][2]; // Increased stack size
20 int top = -1;
```

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

Print Table

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

First, note that