

## EXPERIMENT-1

**PROGRAM:**

```
#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX_LEN 100

void checkToken(char *token) {

    if (isdigit(token[0])) {

        printf("CONSTANT: %s\n", token);

    } else if (isalpha(token[0]) || token[0] == '_') {

        printf("IDENTIFIER: %s\n", token);

    } else if (strchr("+-*/=<>!", token[0])) {

        printf("OPERATOR: %s\n", token);

    }

}

int main() {

    char input[MAX_LEN], buffer[MAX_LEN];

    int i = 0;

    printf("Enter a line of code: ");

    fgets(input, MAX_LEN, stdin);

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

        if (isspace(input[j])) {

            if (i != 0) {

                buffer[i] = '\0';

                checkToken(buffer);

                i = 0;

            }

        } else if (strchr("+-*/=<>!", input[j])) {

            if (i != 0) {
```

```

        buffer[i] = '\0';

        checkToken(buffer);

        i = 0;
    }

    printf("OPERATOR: %c\n", input[j]);
} else {

    buffer[i++] = input[j];

}

return 0;
}

```

### OUTPUT:

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features the Programiz logo, a banner with the text "Turn your knowledge into new customers", and a Google Ads logo. Below the banner is a "Programiz PRO" button. The main content area displays a C code editor with the following code:

```

main.c
1 #include <stdio.h>
2 #include <ctype.h>
3 #include <string.h>
4
5 #define MAX_LEN 100
6
7 void checkToken(char *token) {
8     if (isdigit(token[0])) {
9         printf("CONSTANT: %s\n", token);
10    } else if (isalpha(token[0]) || token[0] ==
11        '_' ) {
12        printf("IDENTIFIER: %s\n", token);
13    } else if (strchr("+-*/=<>!", token[0])) {
14        printf("OPERATOR: %s\n", token);
15    }
16 }

```

Below the code editor is a "Run" button. To the right of the code editor is an "Output" window. The output window shows the following text:

```

Enter a line of code: x=10+y
IDENTIFIER: x
OPERATOR: =
CONSTANT: 10
OPERATOR: +
IDENTIFIER: y

=== Code Execution Successful ===

```

The bottom of the screenshot shows a Windows taskbar with various icons, including the Start button, Search bar, and system tray icons for NIFTY, ENG US, and the date/time (13:34:38, 13-02-2025).

### EXPERIMENT-2

#### PROGRAM:

```

#include <stdio.h>

#include <string.h>

void checkComment(char *line) {

    if (strstr(line, "//") == line) {

```

```

        printf("Single-line comment detected.\n");
    } else if (strstr(line, "/*") == line) {
        printf("Multi-line comment detected.\n");
    } else {
        printf("Not a comment.\n");
    }
}

int main() {
    char line[256];

    printf("Enter a line: ");

    fgets(line, sizeof(line), stdin);

    checkComment(line);

    return 0;
}

```

## OUTPUT:

The screenshot shows the Programiz Online C Compiler interface. The code editor on the left contains the following C code:

```

1 #include <stdio.h>
2 #include <string.h>
3
4 void checkComment(char *line) {
5     if (strstr(line, "//") == line) {
6         printf("Single-line comment detected.\n");
7     } else if (strstr(line, "/*") == line) {
8         printf("Multi-line comment detected.\n");
9     } else {
10        printf("Not a comment.\n");
11    }
12 }
13
14 int main() {
15     char line[256];

```

The output window on the right shows the following output:

```

Enter a line: /*comment*/
Multi-line comment detected.

=== Code Execution Successful ===

```

## EXPERIMENT-3

### PROGRAM:

```

#include <stdio.h>

#include <string.h>

```

```

void checkOperator(char ch) {
    if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
        printf("Operator: %c\n", ch);
    }
}

```

```

int main() {
    char input[100];
    printf("Enter an expression: ");
    fgets(input, sizeof(input), stdin);

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

    return 0;
}

```

### OUTPUT:

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features a blue header with the Programiz logo and a banner for "Premium Coding Courses by Programiz". Below the header, there is a code editor with a file named `main.c`. The code in the editor is as follows:

```

7  }
8  }
9
10 int main() {
11     char input[100];
12     printf("Enter an expression: ");
13     fgets(input, sizeof(input), stdin);
14
15     for (int i = 0; input[i] != '\0'; i++) {
16         checkOperator(input[i]);
17     }
18
19     return 0;
20 }
21

```

To the right of the code editor is an "Output" panel. It displays the following text:

```

Enter an expression: Enter an expression: a + b - c * d /
e

Operator: +
Operator: -
Operator: *
Operator: /

=== Code Execution Successful ===

```

The bottom of the screenshot shows a Windows taskbar with various application icons and a system clock indicating the time is 13:50:15 on 13-02-2025.

#### EXPERIMENT-4

##### PROGRAM:

```
#include <stdio.h>

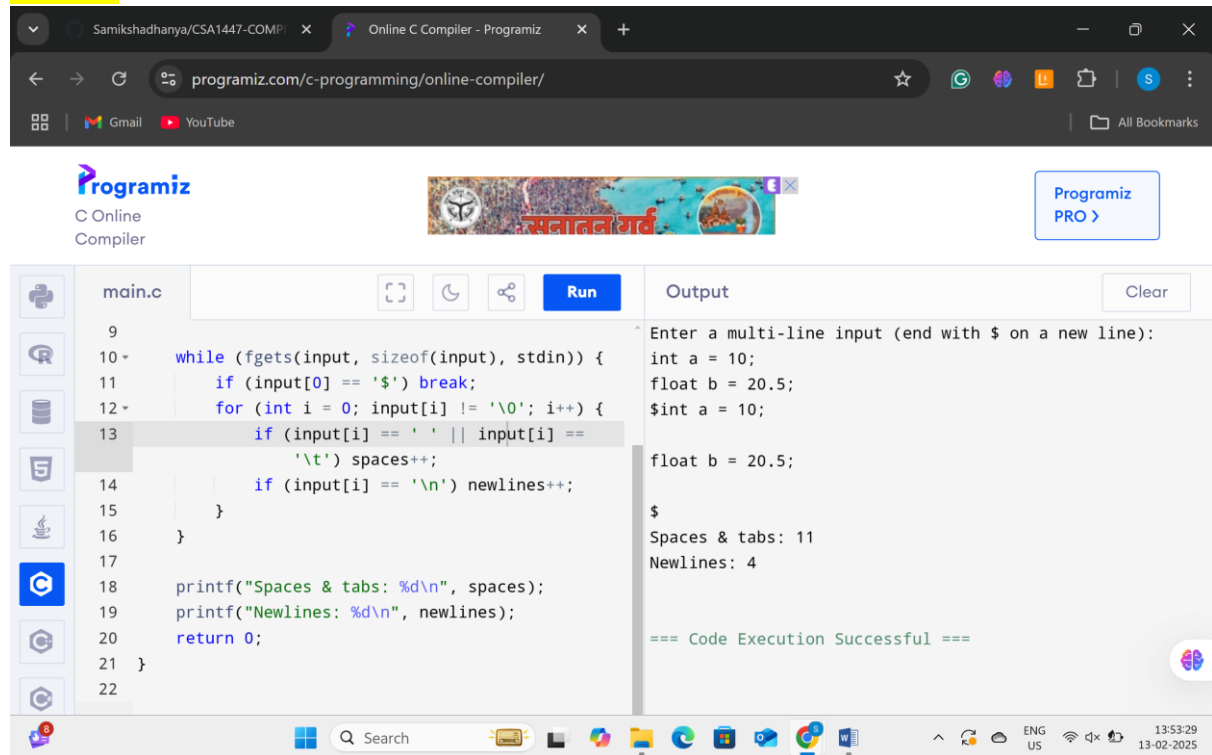
#include <string.h>

int main() {
    char input[200];
    int spaces = 0, newlines = 0;

    printf("Enter a multi-line input (end with $ on a new line):\n");

    while (fgets(input, sizeof(input), stdin)) {
        if (input[0] == '$') break;
        for (int i = 0; input[i] != '\0'; i++) {
            if (input[i] == ' ' || input[i] == '\t') spaces++;
            if (input[i] == '\n') newlines++;}
        printf("Spaces & tabs: %d\n", spaces);
        printf("Newlines: %d\n", newlines);
    }
    return 0;}
```

## OUTPUT:



The screenshot shows the Programiz Online C Compiler interface. The code editor on the left contains a C program named `main.c` that counts spaces, tabs, and newlines in a multi-line input. The output window on the right shows the program's execution results.

```
main.c
9
10 while (fgets(input, sizeof(input), stdin)) {
11     if (input[0] == '$') break;
12     for (int i = 0; input[i] != '\0'; i++) {
13         if (input[i] == ' ' || input[i] ==
            '\t') spaces++;
14         if (input[i] == '\n') newlines++;
15     }
16 }
17
18 printf("Spaces & tabs: %d\n", spaces);
19 printf("Newlines: %d\n", newlines);
20 return 0;
21 }
22
```

Output

```
Enter a multi-line input (end with $ on a new line):
int a = 10;
float b = 20.5;
$int a = 10;

float b = 20.5;

$
Spaces & tabs: 11
Newlines: 4

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

## EXPERIMENT-5

### PROGRAM:

```
#include <stdio.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[50];
    printf("Enter an identifier: ");
```

```

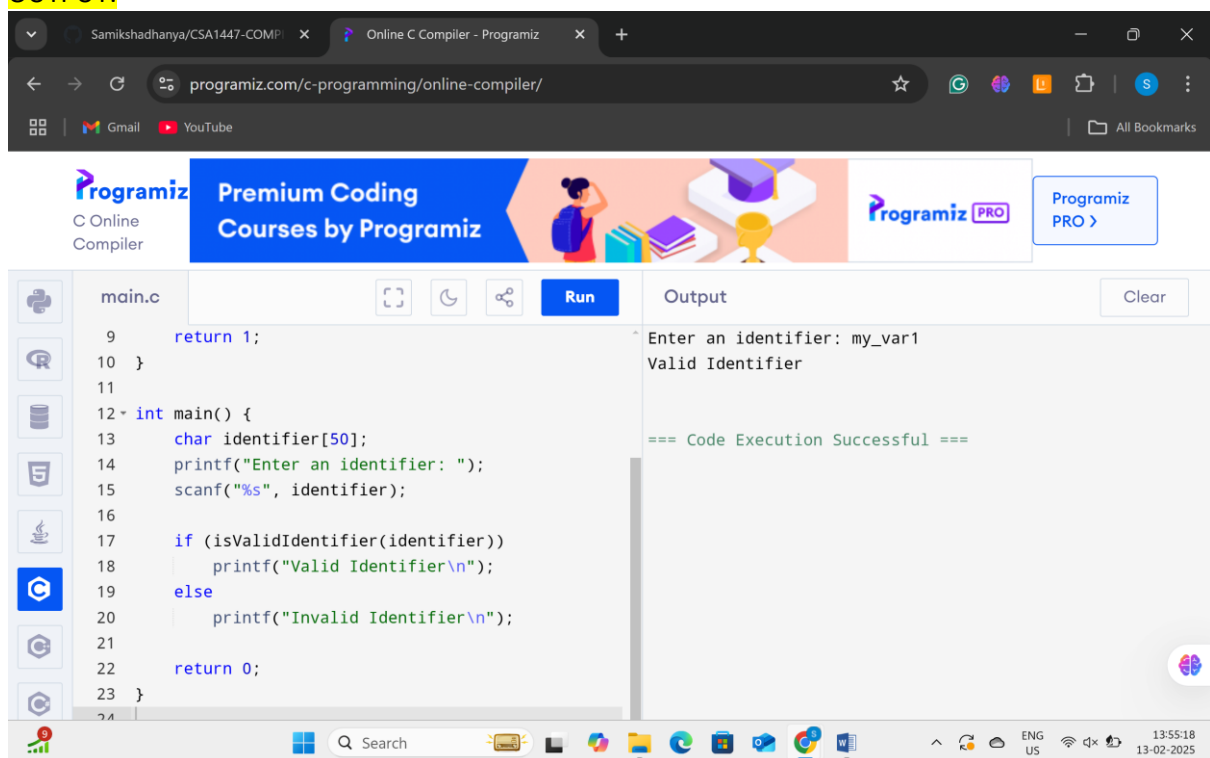
scanf("%s", identifier);

if (isValidIdentifier(identifier))
    printf("Valid Identifier\n");
else
    printf("Invalid Identifier\n");

return 0;
}

```

### OUTPUT:



### EXPERIMENT-6

#### PROGRAM:

```

#include <stdio.h>

#include <string.h>

void removeLeftRecursion(char nonTerminal, char alpha[], char beta[]) {

    char newNonTerminal = nonTerminal + '1';

    printf("%c -> %s%c1\n", nonTerminal, beta, newNonTerminal);

    printf("%c1 -> %s%c1 | ε\n", newNonTerminal, alpha, newNonTerminal);
}

int main() {

```

```

char nonTerminal, alpha[10], beta[10];

printf("Enter production (A->A $\alpha$ | $\beta$ ): ");

scanf(" %c->%[^|]|%s", &nonTerminal, alpha, beta);

if (alpha[0] == nonTerminal)

    removeLeftRecursion(nonTerminal, alpha + 1, beta);

else

    printf("No Left Recursion: %c -> %s | %s\n", nonTerminal, alpha, beta);

return 0;}

```

### OUTPUT:

The screenshot shows a web browser at `programiz.com/c-programming/online-compiler/`. The code editor contains the following C code:

```

11 char nonTerminal, alpha[10], beta[10];
12 printf("Enter production (A->A $\alpha$ | $\beta$ ): ");
13 scanf(" %c->%[^|]|%s", &nonTerminal, alpha, beta);
14
15 if (alpha[0] == nonTerminal)
16     removeLeftRecursion(nonTerminal, alpha + 1, beta);
17 else
18     printf("No Left Recursion: %c -> %s | %s\n", nonTerminal, alpha, beta);
19
20 return 0;
21 }
22

```

The output window shows the following text:

```

Enter production (A->A $\alpha$ | $\beta$ ): A->A $\alpha$ | $\beta$ 
A -> br1
r1 -> ar1 |  $\epsilon$ 

=== Code Execution Successful ===

```

### EXPERIMENT-7

#### PROGRAM:

```

#include <stdio.h>

#include <string.h>

void eliminateLeftFactoring(char nt, char alpha[], char beta[]) {

    char newNT = nt + '1';

    printf("%c -> %s%c1\n", nt, alpha, newNT);

    printf("%c1 -> %s |  $\epsilon$ \n", newNT, beta);

}int main() {

```



```

char nt, alpha[10], beta[10];

printf("Enter production (A-> $\alpha\beta_1$  |  $\alpha\beta_2$ ): ");
scanf(" %c->[%^|]|%s", &nt, alpha, beta);

if (strncmp(alpha, beta, 1) == 0)

    eliminateLeftFactoring(nt, alpha, beta);

else

    printf("No Left Factoring: %c -> %s | %s\n", nt, alpha, beta);

return 0;

```

### OUTPUT:

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page features a banner for "Premium Coding Courses by Programiz". Below the banner, the "main.c" file is open in the editor, displaying the C code for checking left factoring. The code prompts the user to enter a production rule in the form `A-> $\alpha\beta_1$  |  $\alpha\beta_2$` . The user has entered `A->abc|abdB->abc|abd`. The program then prompts for a non-terminal, and the user has entered `r1`. The output shows the result `r1 -> abd | ε`, indicating that the first characters of the two right-hand sides do not match, so no left factoring is possible. The status "Code Execution Successful" is displayed at the bottom of the output pane.

```

main.c
9
10 int main() {
11     char nt, alpha[10], beta[10];
12     printf("Enter production (A-> $\alpha\beta_1$  |  $\alpha\beta_2$ ): ");
13     scanf(" %c->[%^|]|%s", &nt, alpha, beta);
14
15     if (strncmp(alpha, beta, 1) == 0)
16         eliminateLeftFactoring(nt, alpha, beta);
17     else
18         printf("No Left Factoring: %c -> %s | %s\n", nt, alpha, beta);
19
20     return 0;
21 }
22
Output
Enter production (A-> $\alpha\beta_1$  |  $\alpha\beta_2$ ): A->abc|abdB->abc|abd
A -> abcr1
r1 -> abd | ε

=== Code Execution Successful ===

```