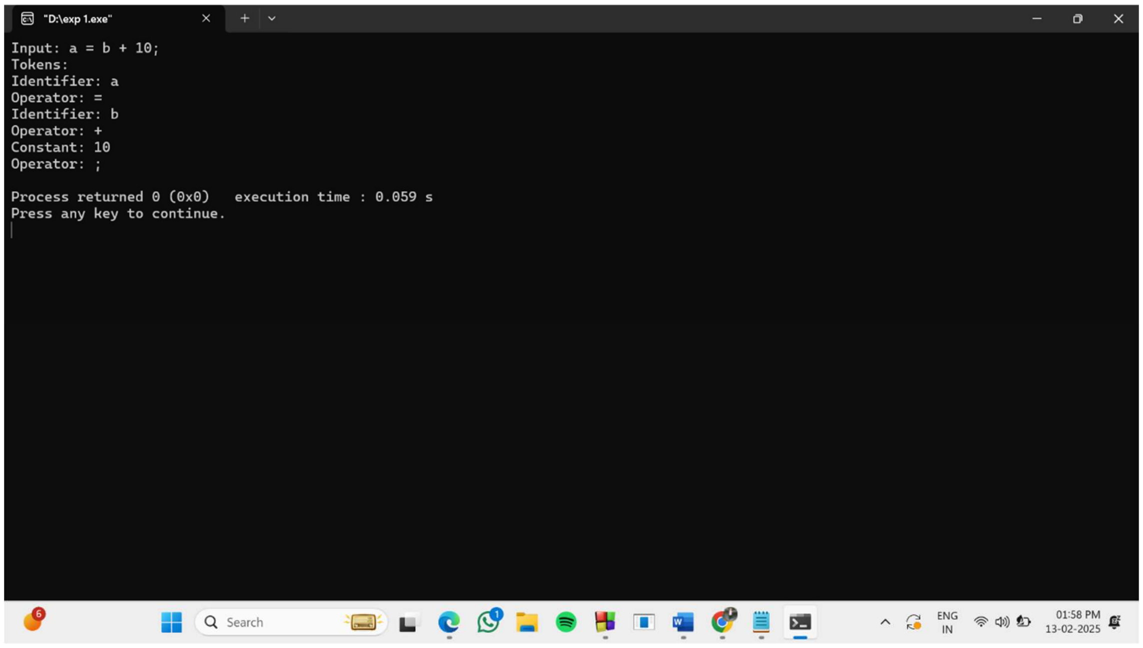
Experiment - 1

```
#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX_LEN 20

int isOperator(char ch) {

    char operators[] = "+-*/=%;";

    for (int i = 0; i < strlen(operators); i++) {

        if (ch == operators[i]) return 1;

    }

    return 0;

}

int isIdentifier(char token[]) {

    if (!isalpha(token[0]) && token[0] != '_') return 0;

    for (int i = 1; token[i] != '\0'; i++) {

        if (!isalnum(token[i]) && token[i] != '_') return 0;

    }

    return 1;

}

int isNumber(char token[]) {

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

        if (!isdigit(token[i])) return 0;

    }

    return 1;

}

int main() {

    char input[] = "a = b + 10;";

    char token[MAX_LEN];

    int i = 0, j = 0;

    printf("Input: %s\n", input);
    printf("Tokens:\n");
  while (input[i] != '\0') {
        if (isOperator(input[i])) {
            printf("Operator: %c\n", input[i]);
        } else if (isalnum(input[i]) || input[i] == '_') {
            j = 0;
            while (isalnum(input[i]) || input[i] == '_') {
                if (j < MAX_LEN - 1) {
                    token[j++] = input[i];
                }
                i++;
            }
            token[j] = '\0';
            i--;
            if (isNumber(token))
                printf("Constant: %s\n", token);
            else if (isIdentifier(token))
                printf("Identifier: %s\n", token);
        }
        i++;
    }
    return 0;
```

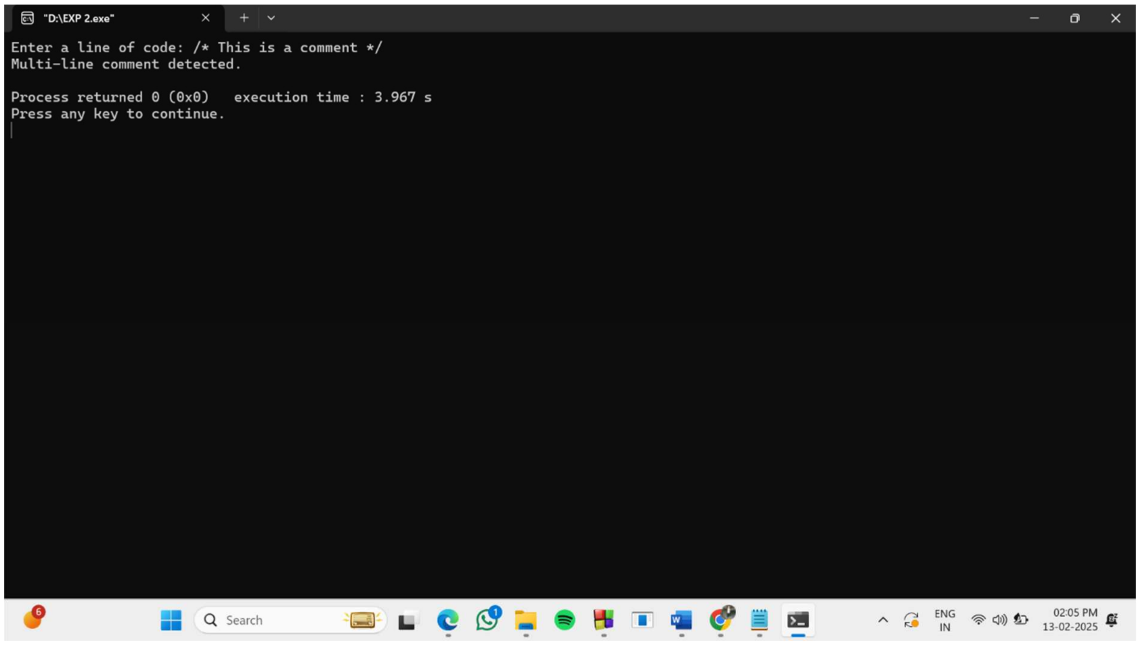
}



```
Input: a = b + 10;
Tokens:
Identifier: a
Operator: =
Identifier: b
Operator: +
Constant: 10
Operator: ;

Process returned 0 (0x0)   execution time : 0.059 s
Press any key to continue.
```

Experiment 2

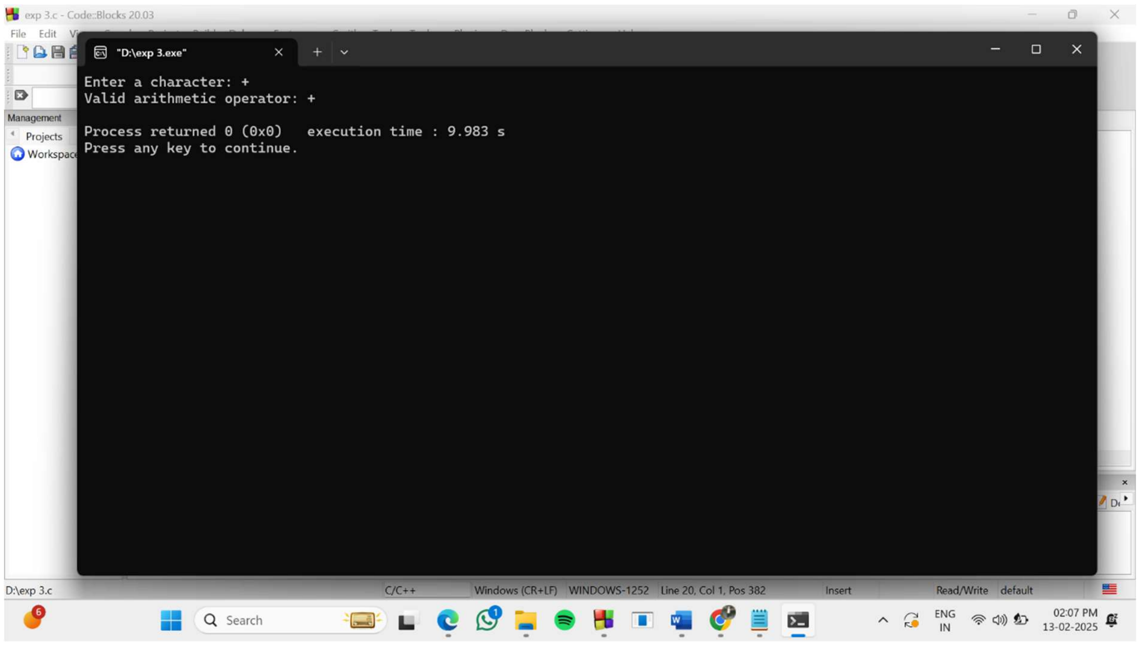
```
#include <stdio.h>

#include <string.h>

void check_comment(char *line) {

    if (strncmp(line, "//", 2) == 0) {

        printf("Single-line comment detected.\n");

    } else if (strncmp(line, "/*", 2) == 0 && strstr(line, "*/") != NULL) {

        printf("Multi-line comment detected.\n");

    } else {

        printf("Not a comment.\n");

    }

}

int main() {

    char line[256];

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

    fgets(line, sizeof(line), stdin);
    check_comment(line);
    return 0;
}
```



```
Enter a line of code: /* This is a comment */
Multi-line comment detected.

Process returned 0 (0x0)   execution time : 3.967 s
Press any key to continue.
```

Experiment – 3

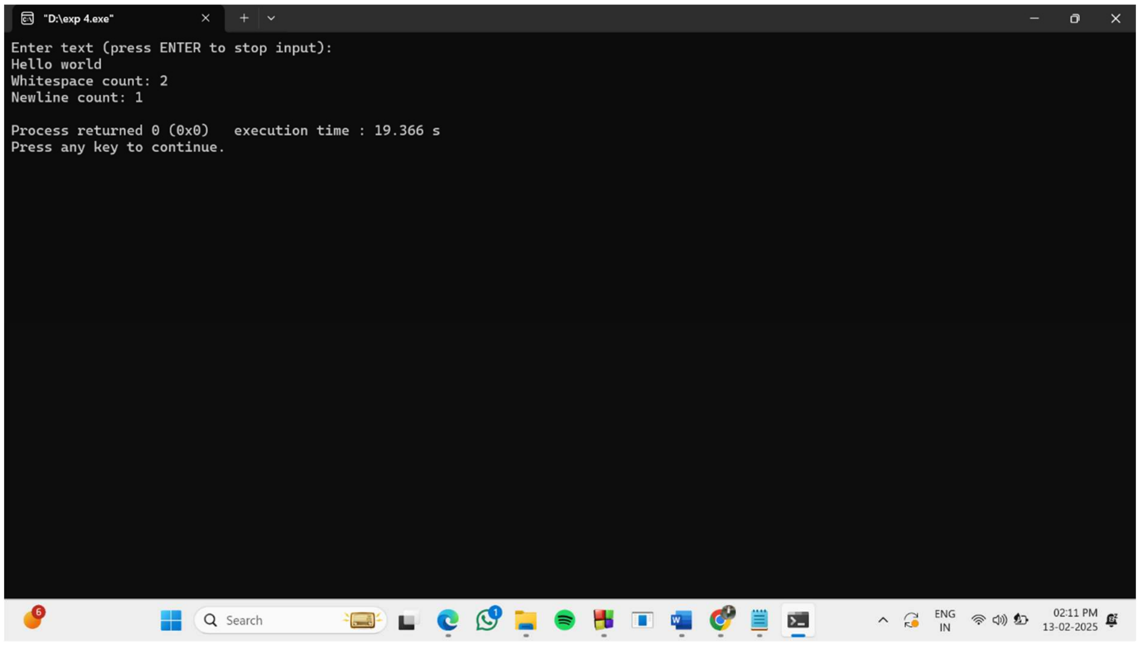
```
#include <stdio.h>
void check_operator(char ch) {
    if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
        printf("Valid arithmetic operator: %c\n", ch);
    } else {
        printf("Not an arithmetic operator.\n");
    }
}
int main() {
    char ch
    printf("Enter a character: ");
    scanf(" %c", &ch);
    check_operator(ch);
    return 0;}
```



```
Enter a character: +
Valid arithmetic operator: +

Process returned 0 (0x0)   execution time : 9.983 s
Press any key to continue.
```

Experiment – 4

```
#include <stdio.h>

#include <string.h>

int main() {

    char text[256];

    int whitespace = 0, newline = 0;

    printf("Enter text (press ENTER to stop input):\n");

    fgets(text, sizeof(text), stdin);

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

        if (text[i] == ' ' || text[i] == '\t')

            whitespace++;

        else if (text[i] == '\n')

            newline++;

    }

    printf("Whitespace count: %d\n", whitespace);

    printf("Newline count: %d\n", newline);

    return 0;

}
```

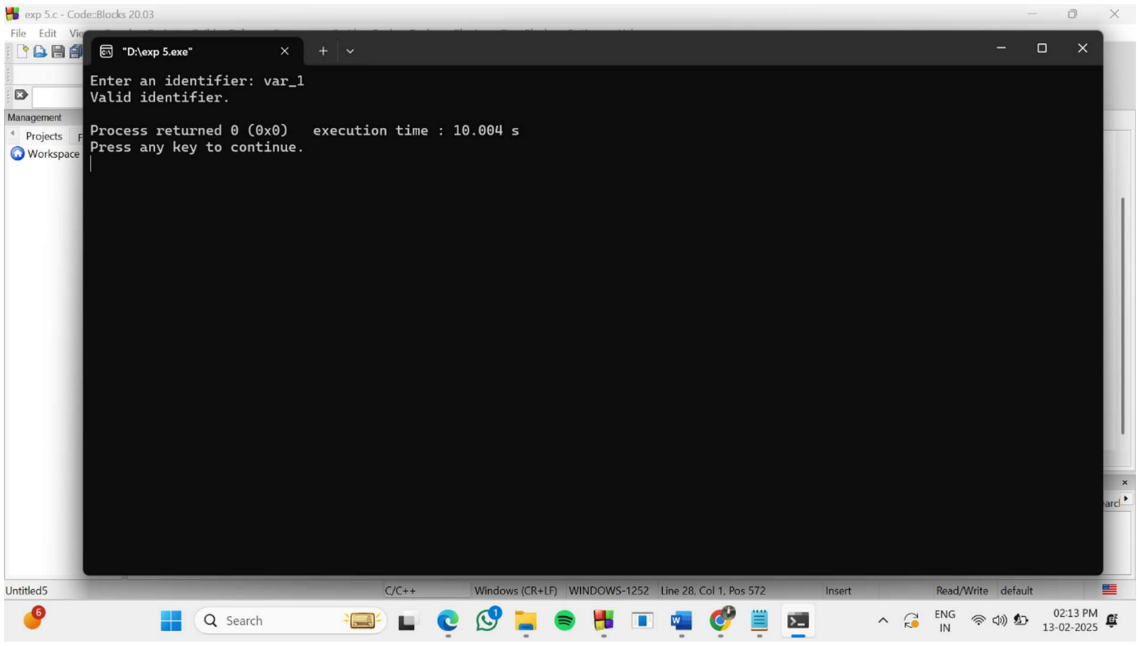


```
Enter text (press ENTER to stop input):
Hello world
Whitespace count: 2
Newline count: 1

Process returned 0 (0x0)   execution time : 19.366 s
Press any key to continue.
```

Experiment – 5

```
#include <stdio.h>

#include <ctype.h>

#include <string.h>

int is_valid_identifier(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 (is_valid_identifier(identifier))

      printf("Valid identifier.\n");
```

```
  else
    printf("Invalid identifier.\n");
  return 0;
}
```

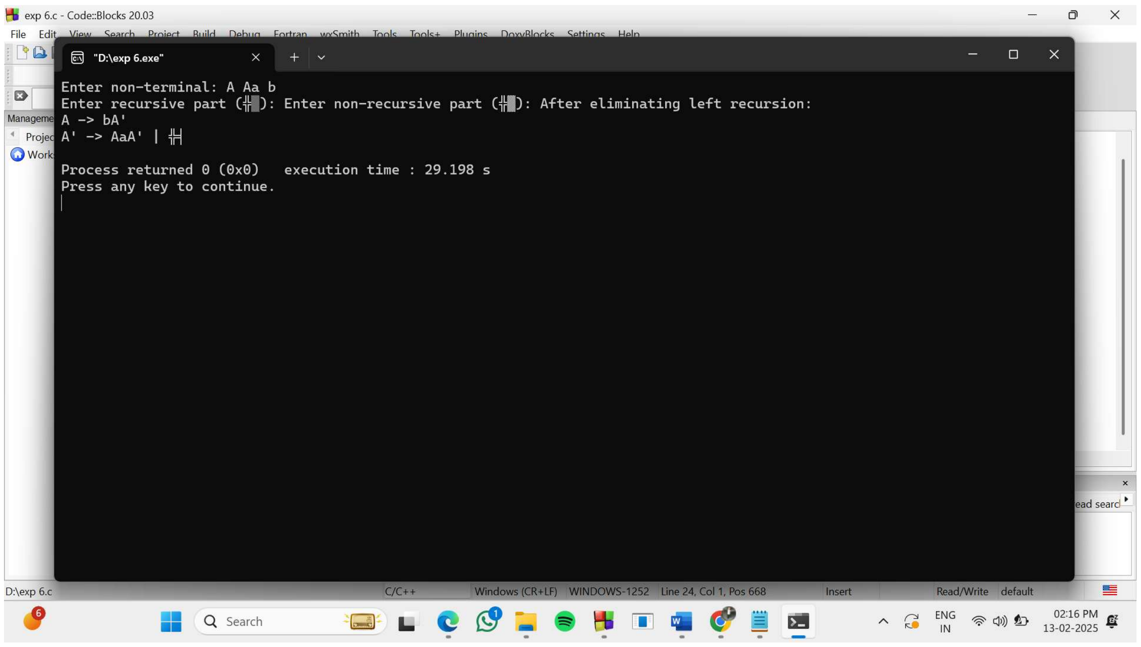


```
Enter an identifier: var_1
Valid identifier.

Process returned 0 (0x0)   execution time : 10.004 s
Press any key to continue.
```

Experiment – 6

```
#include <stdio.h>
#include <string.h>
void eliminate_left_recursion(char *non_terminal, char *alpha, char *beta) {
  printf("After eliminating left recursion:\n");
  printf("%s -> %s%s'\n", non_terminal, beta, non_terminal);
  printf("%s' -> %s%s' | ε\n", non_terminal, alpha, non_terminal);
}
int main() {
  char non_terminal[10], alpha[10], beta[10];
  printf("Enter non-terminal: ");
  scanf("%s", non_terminal);
  printf("Enter recursive part (α): ");
  scanf("%s", alpha);
  printf("Enter non-recursive part (β): ");
```

```
    scanf("%s", beta);
    eliminate_left_recursion(non_terminal, alpha, beta);
    return 0;
}
```

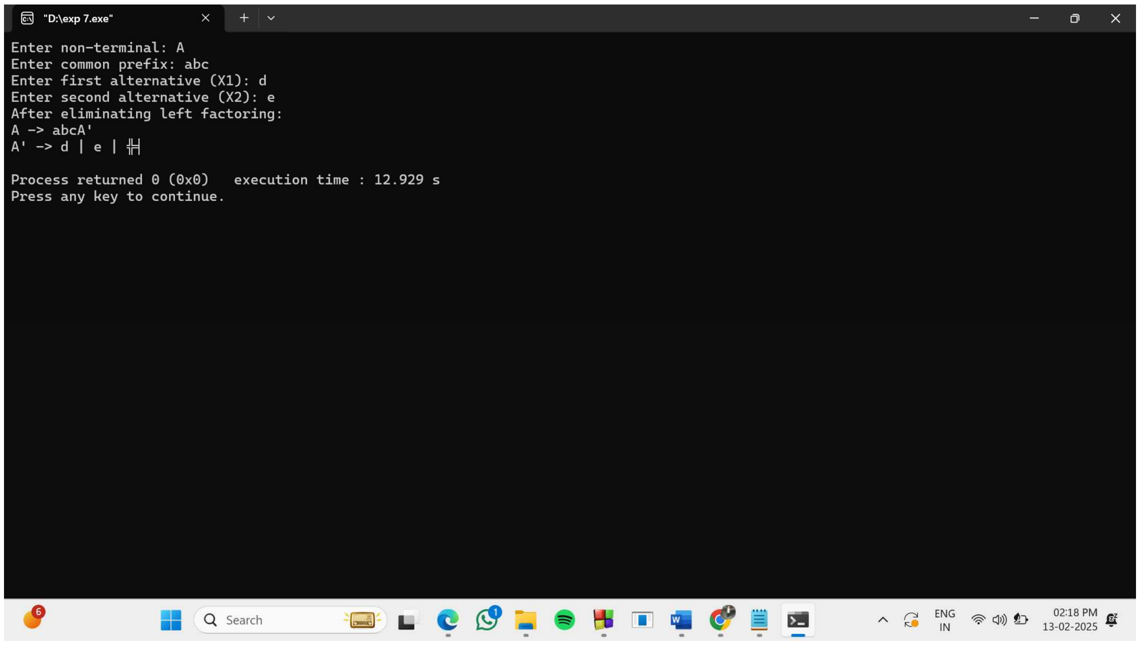


```
Enter non-terminal: A Aa b
Enter recursive part (╬▒): Enter non-recursive part (╬▓): After eliminating left recursion:
A -> bA'
A' -> AaA' | ╬╡

Process returned 0 (0x0)   execution time : 29.198 s
Press any key to continue.
```

Experiment – 7

```
#include <stdio.h>
#include <string.h>
void eliminate_left_factoring(char *non_terminal, char *common, char *x1, char *x2) {
    printf("After eliminating left factoring:\n");
    printf("%s -> %s%s'\n", non_terminal, common, non_terminal);
    printf("%s' -> %s | %s | ε\n", non_terminal, x1, x2);
}
int main() {
    char non_terminal[10], common[10], x1[10], x2[10];
    printf("Enter non-terminal: ");
    scanf("%s", non_terminal);
    printf("Enter common prefix: ");
    scanf("%s", common);
    printf("Enter first alternative (X1): ");
```

```c
    scanf("%s", x1);

    printf("Enter second alternative (X2): ");

    scanf("%s", x2);

    eliminate_left_factoring(non_terminal, common, x1, x2);

    return 0;

}
```



"D:\exp 7.exe"

```
Enter non-terminal: A
Enter common prefix: abc
Enter first alternative (X1): d
Enter second alternative (X2): e
After eliminating left factoring:
A -> abcA'
A' -> d | e | ╫

Process returned 0 (0x0)   execution time : 12.929 s
Press any key to continue.
```