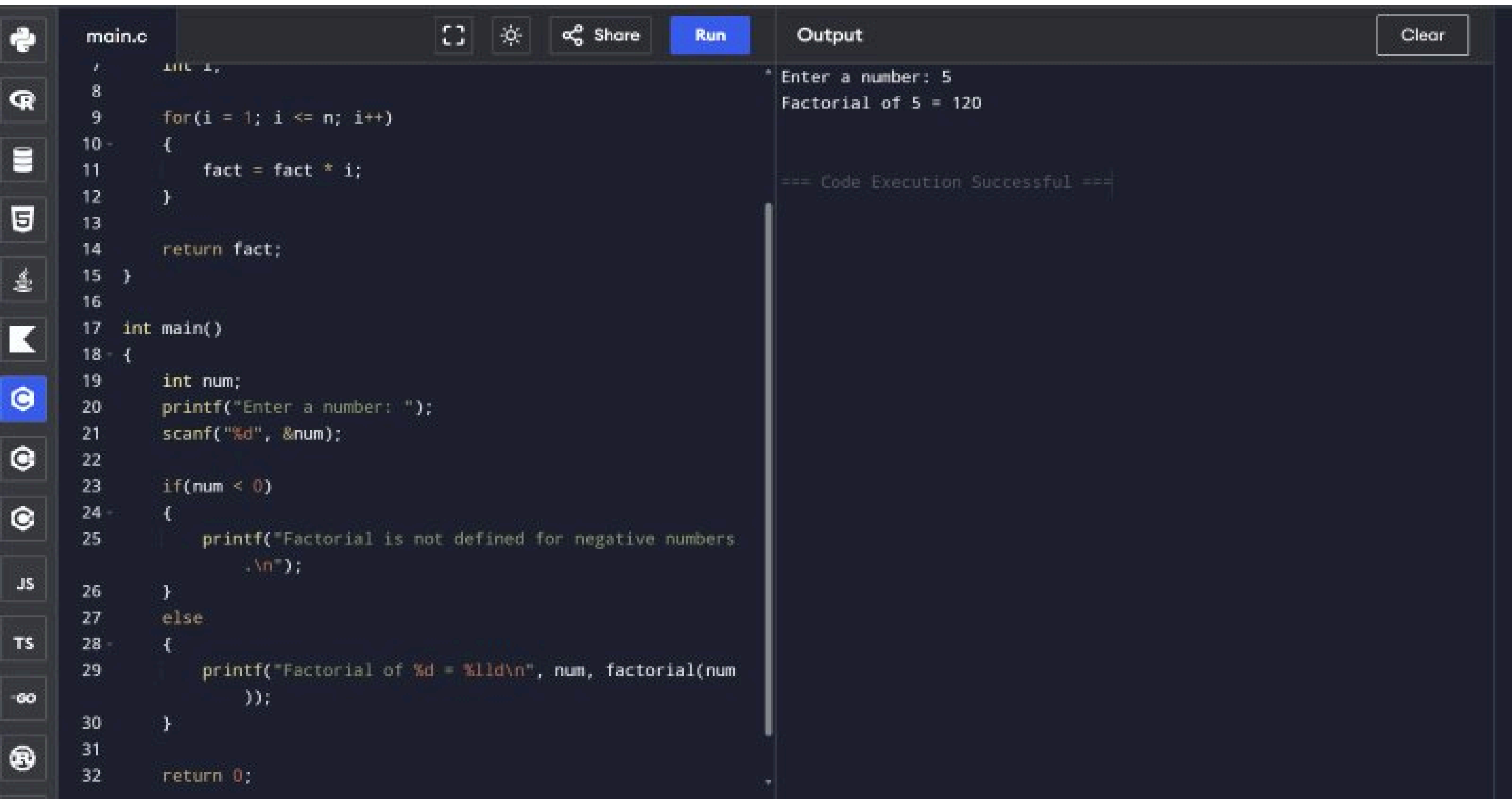
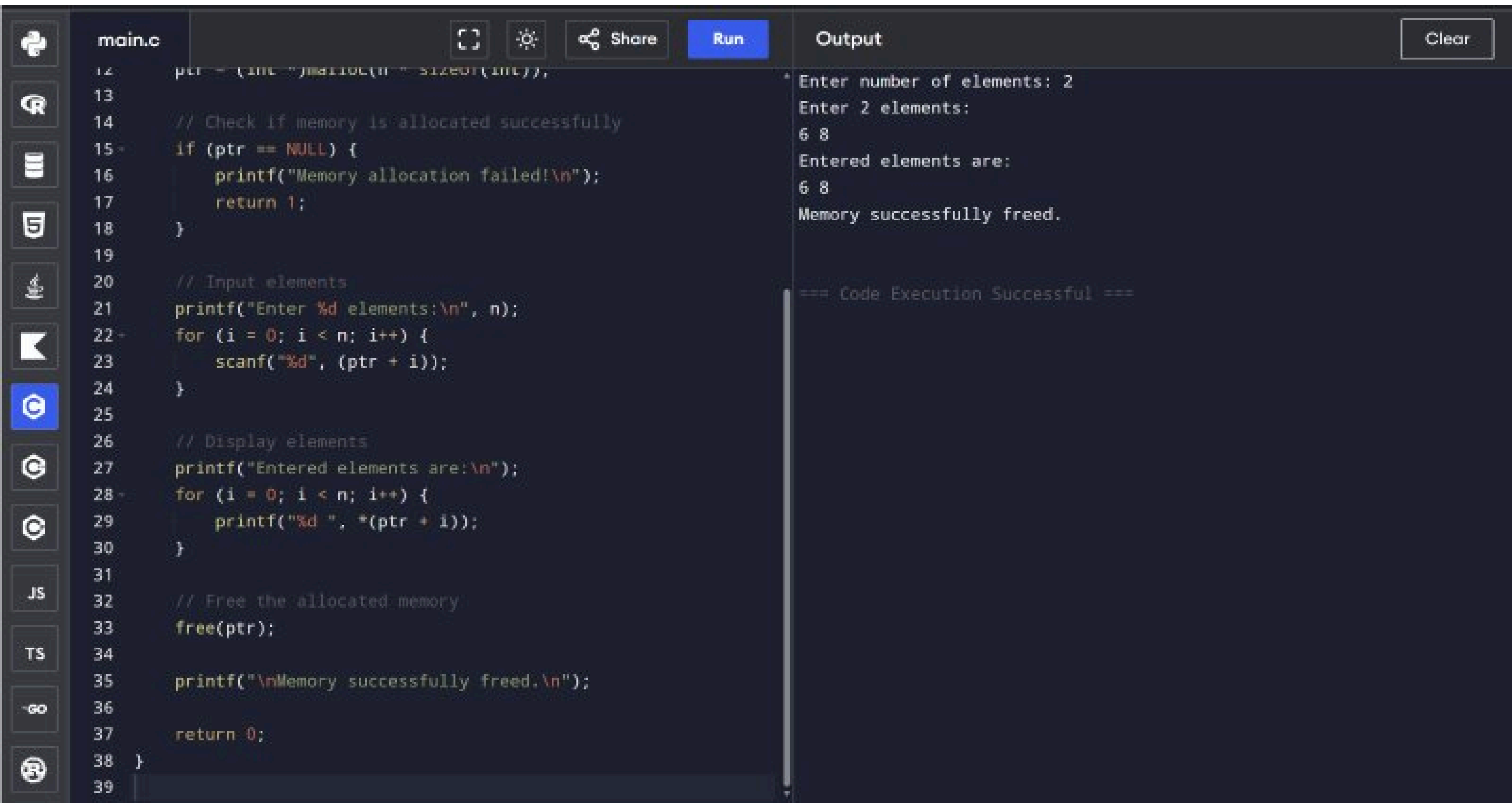


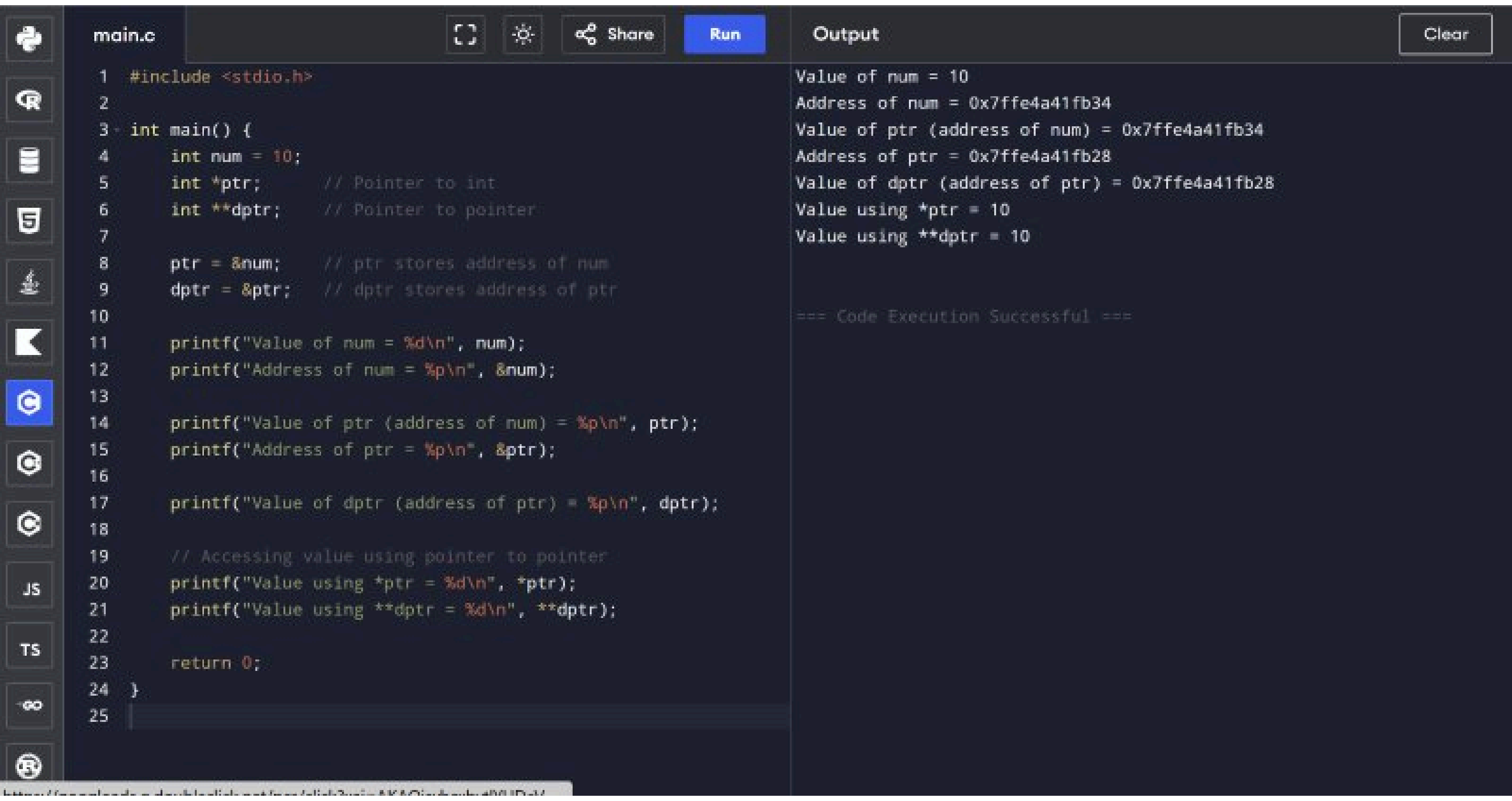
```c
    int i;

    for(i = 1; i <= n; i++)
    {
        fact = fact * i;
    }

    return fact;
}

int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    if(num < 0)
    {
        printf("Factorial is not defined for negative numbers.\n");
    }
    else
    {
        printf("Factorial of %d = %lld\n", num, factorial(num));
    }

    return 0;
```

```
Enter a number: 5
Factorial of 5 = 120


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



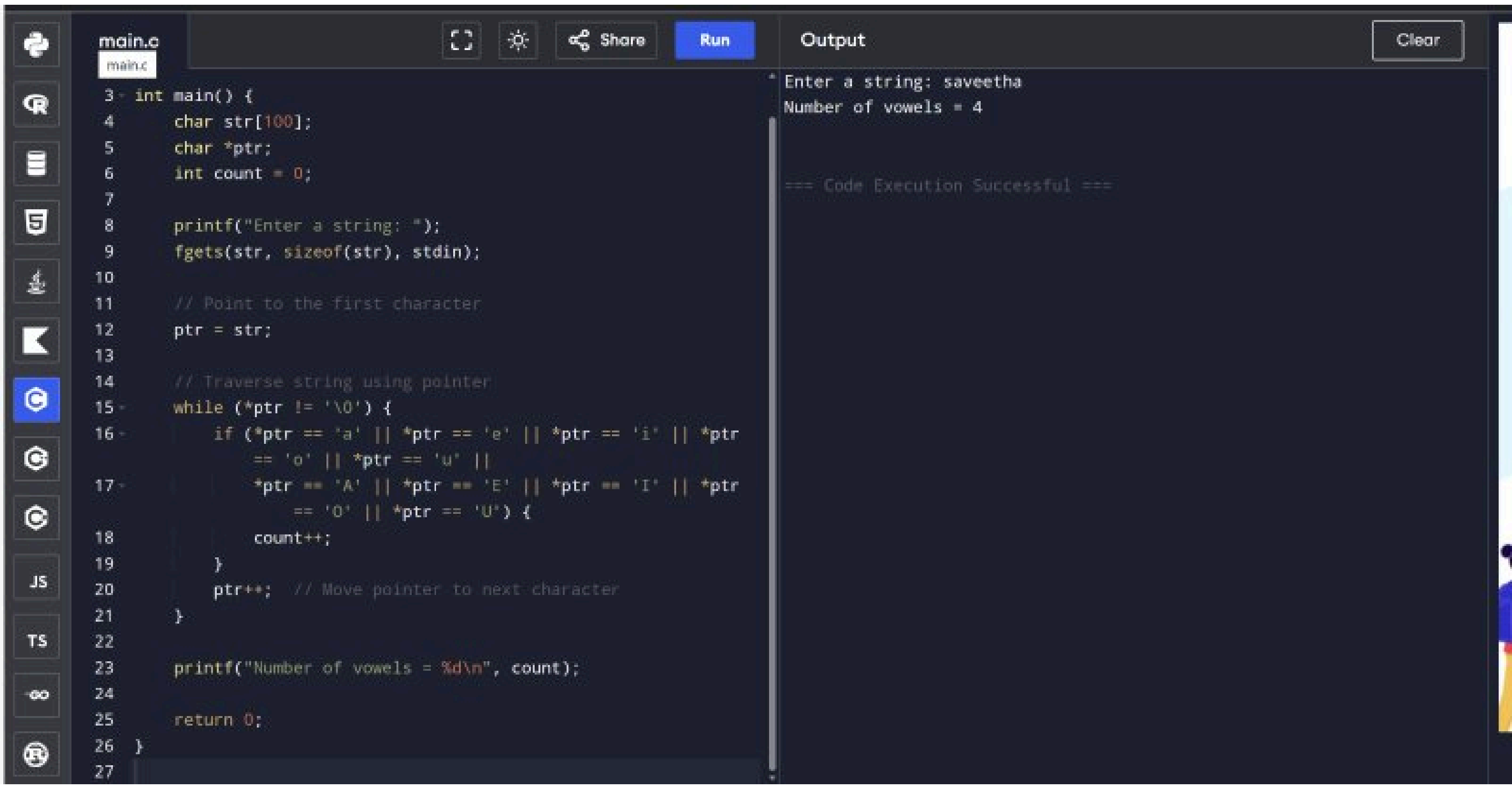
```c
    ptr = (int *)malloc(n * sizeof(int));

    // Check if memory is allocated successfully
    if (ptr == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }

    // Input elements
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", (ptr + i));
    }

    // Display elements
    printf("Entered elements are:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", *(ptr + i));
    }

    // Free the allocated memory
    free(ptr);

    printf("\nMemory successfully freed.\n");

    return 0;
}
```

```
Enter number of elements: 2
Enter 2 elements:
6 8
Entered elements are:
6 8
Memory successfully freed.


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



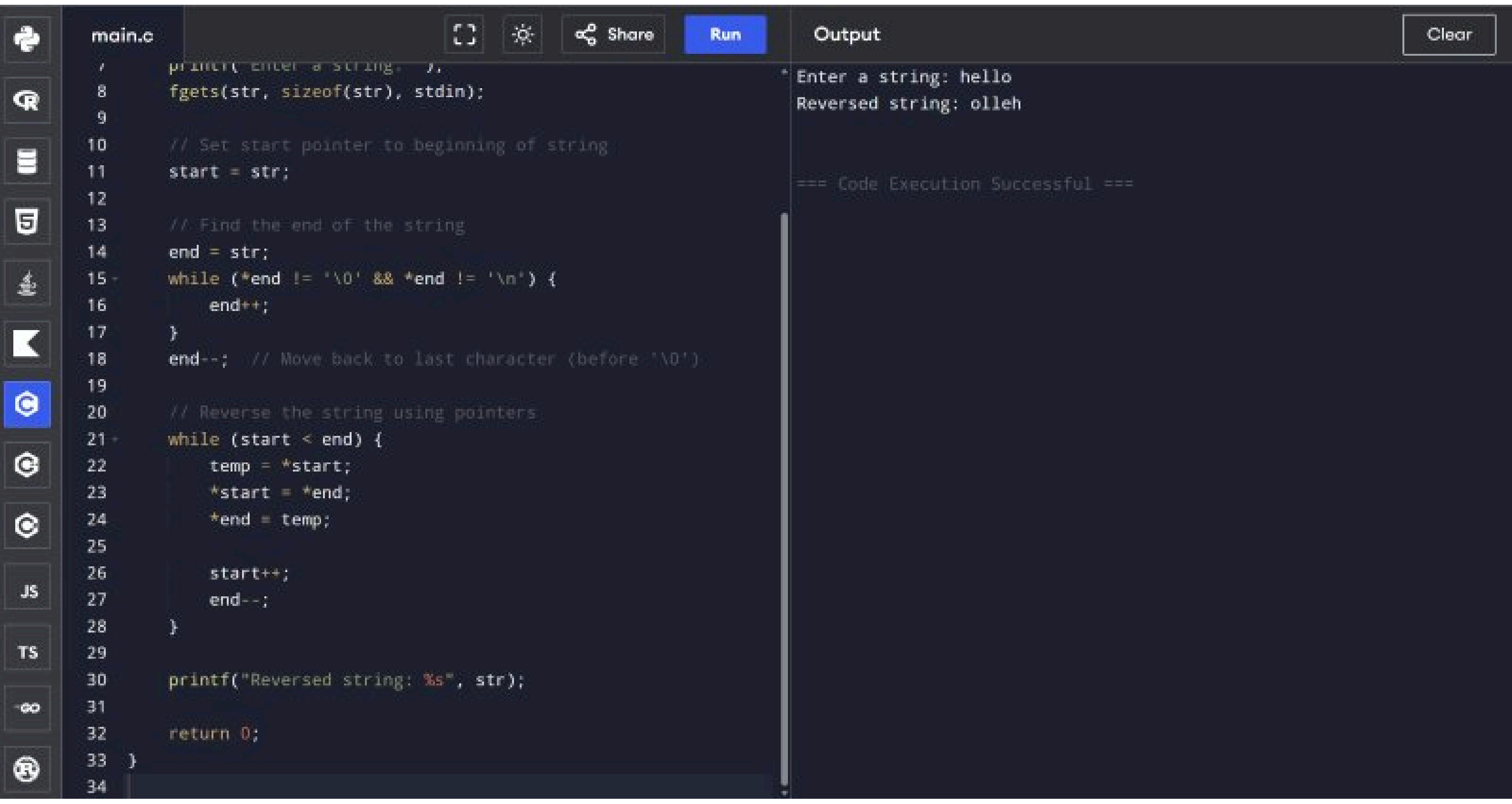
```c
#include <stdio.h>

int main() {
    int num = 10;
    int *ptr;      // Pointer to int
    int **dptr;    // Pointer to pointer

    ptr = &num;    // ptr stores address of num
    dptr = &ptr;   // dptr stores address of ptr

    printf("Value of num = %d\n", num);
    printf("Address of num = %p\n", &num);

    printf("Value of ptr (address of num) = %p\n", ptr);
    printf("Address of ptr = %p\n", &ptr);

    printf("Value of dptr (address of ptr) = %p\n", dptr);

    // Accessing value using pointer to pointer
    printf("Value using *ptr = %d\n", *ptr);
    printf("Value using **dptr = %d\n", **dptr);

    return 0;
}

```

```
Value of num = 10
Address of num = 0x7ffe4a41fb34
Value of ptr (address of num) = 0x7ffe4a41fb34
Address of ptr = 0x7ffe4a41fb28
Value of dptr (address of ptr) = 0x7ffe4a41fb28
Value using *ptr = 10
Value using **dptr = 10


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



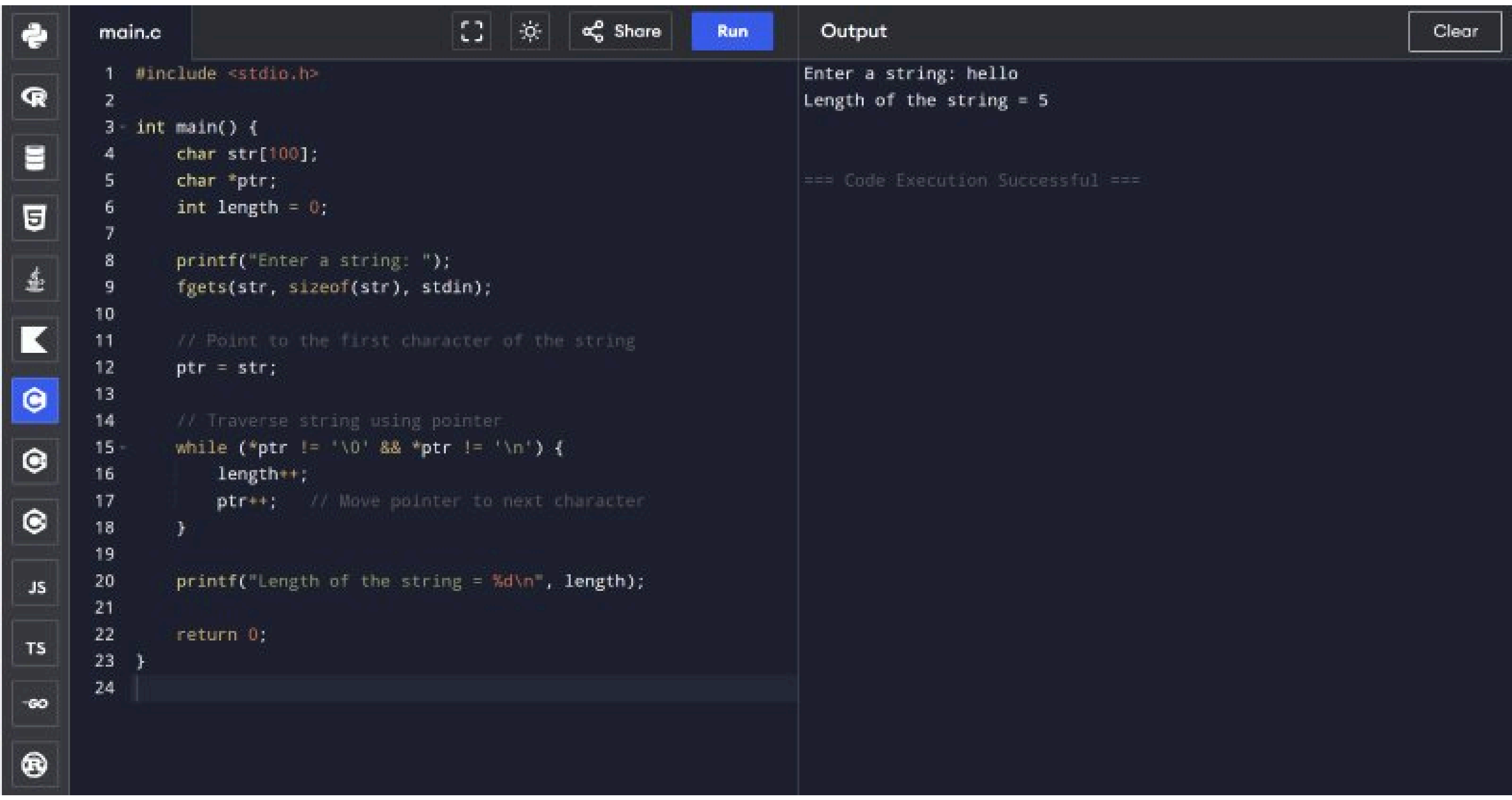
```c
int main() {
    char str[100];
    char *ptr;
    int count = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Point to the first character
    ptr = str;

    // Traverse string using pointer
    while (*ptr != '\0') {
        if (*ptr == 'a' || *ptr == 'e' || *ptr == 'i' || *ptr
            == 'o' || *ptr == 'u' ||
            *ptr == 'A' || *ptr == 'E' || *ptr == 'I' || *ptr
                == 'O' || *ptr == 'U') {
            count++;
        }
        ptr++;  // Move pointer to next character
    }

    printf("Number of vowels = %d\n", count);

    return 0;
}
```

```
Enter a string: saveetha
Number of vowels = 4


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



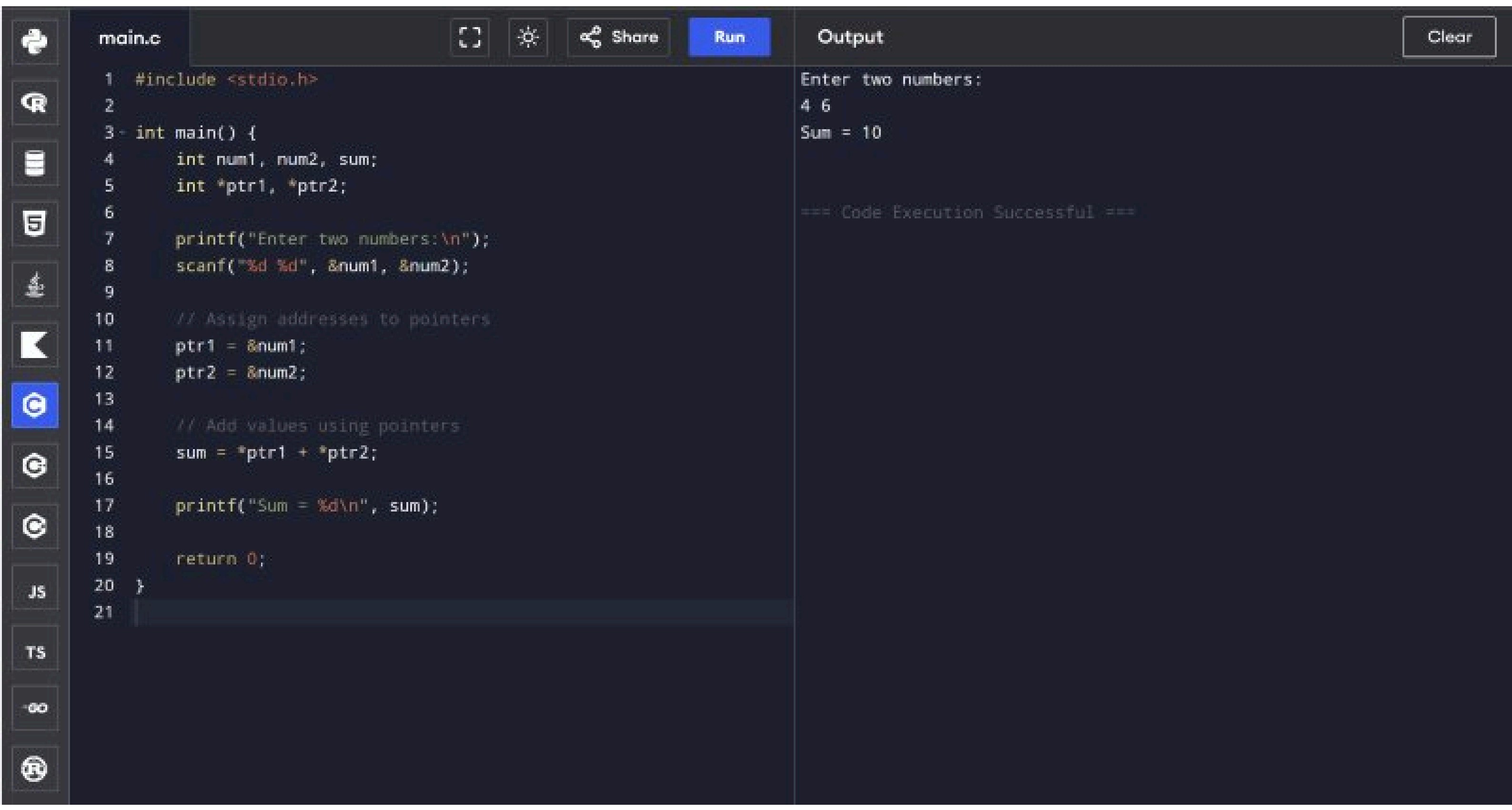
```c
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Set start pointer to beginning of string
    start = str;

    // Find the end of the string
    end = str;
    while (*end != '\0' && *end != '\n') {
        end++;
    }
    end--;  // Move back to last character (before '\0')

    // Reverse the string using pointers
    while (start < end) {
        temp = *start;
        *start = *end;
        *end = temp;

        start++;
        end--;
    }

    printf("Reversed string: %s", str);

    return 0;
}
```

```
Enter a string: hello
Reversed string: olleh


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



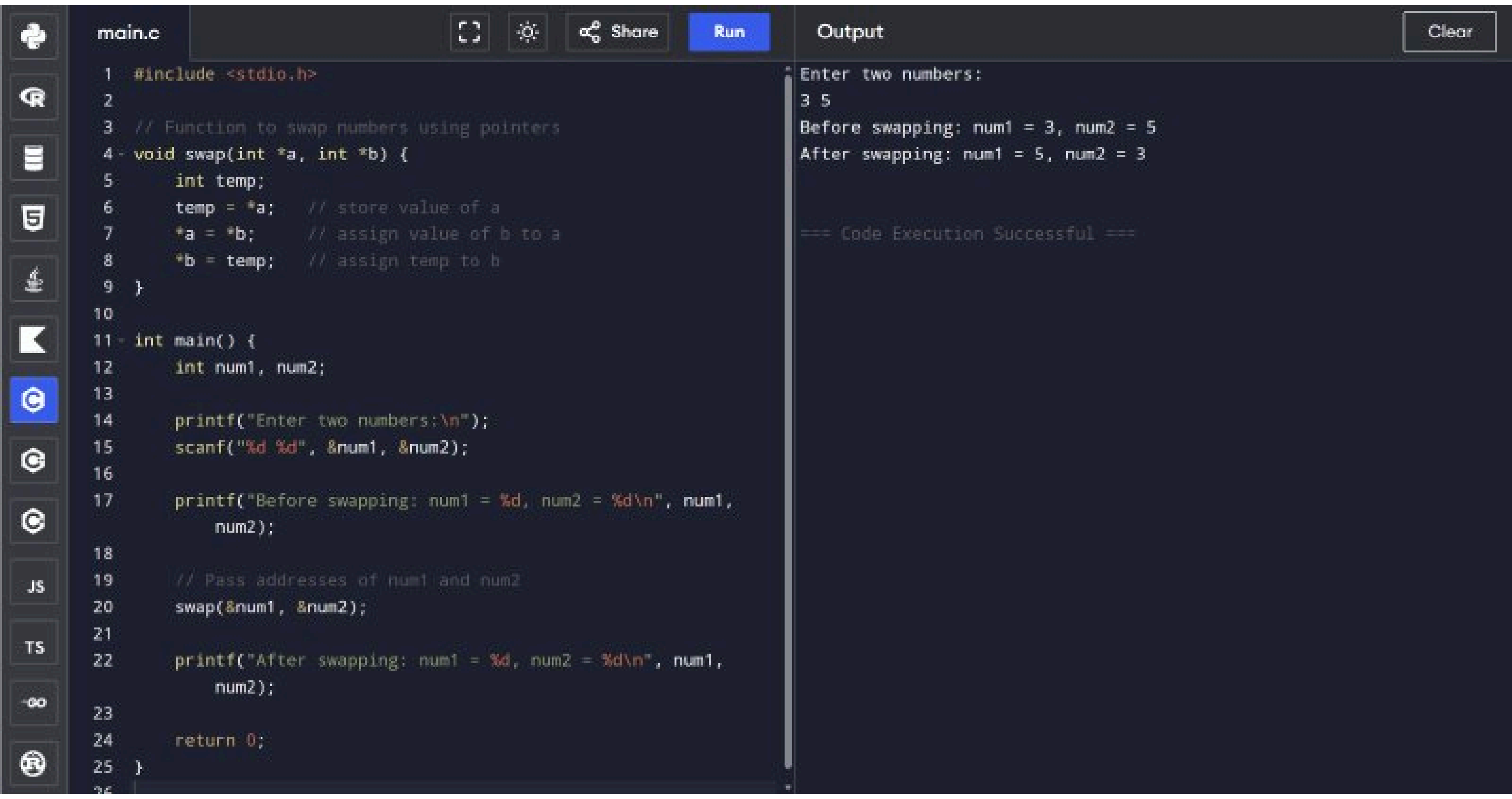
```c
#include <stdio.h>

int main() {
    char str[100];
    char *ptr;
    int length = 0;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Point to the first character of the string
    ptr = str;

    // Traverse string using pointer
    while (*ptr != '\0' && *ptr != '\n') {
        length++;
        ptr++;   // Move pointer to next character
    }

    printf("Length of the string = %d\n", length);

    return 0;
}
```

```
Enter a string: hello
Length of the string = 5


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



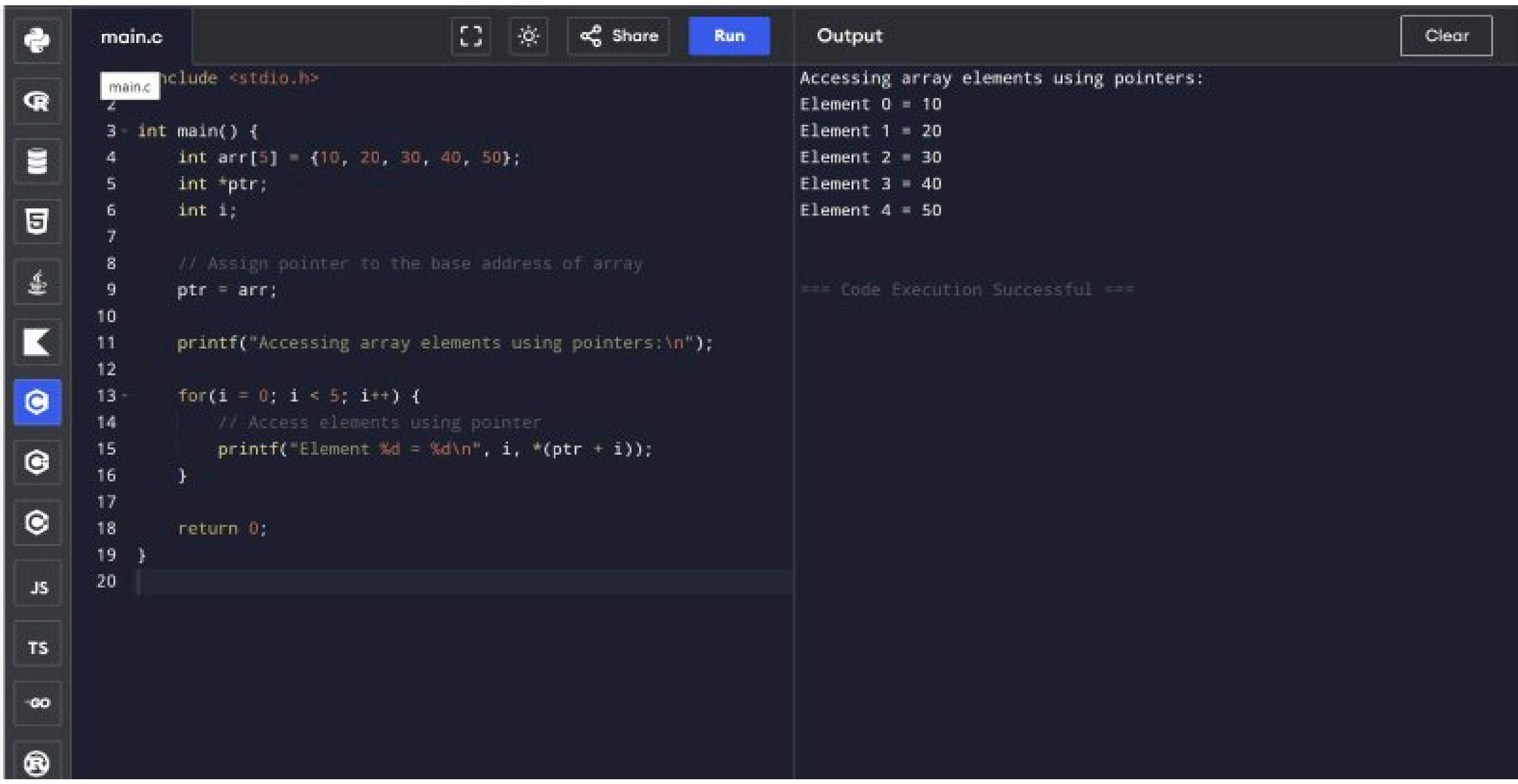
```c
#include <stdio.h>

int main() {
    int num1, num2, sum;
    int *ptr1, *ptr2;

    printf("Enter two numbers:\n");
    scanf("%d %d", &num1, &num2);

    // Assign addresses to pointers
    ptr1 = &num1;
    ptr2 = &num2;

    // Add values using pointers
    sum = *ptr1 + *ptr2;

    printf("Sum = %d\n", sum);

    return 0;
}
```

```
Enter two numbers:
4 6
Sum = 10


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



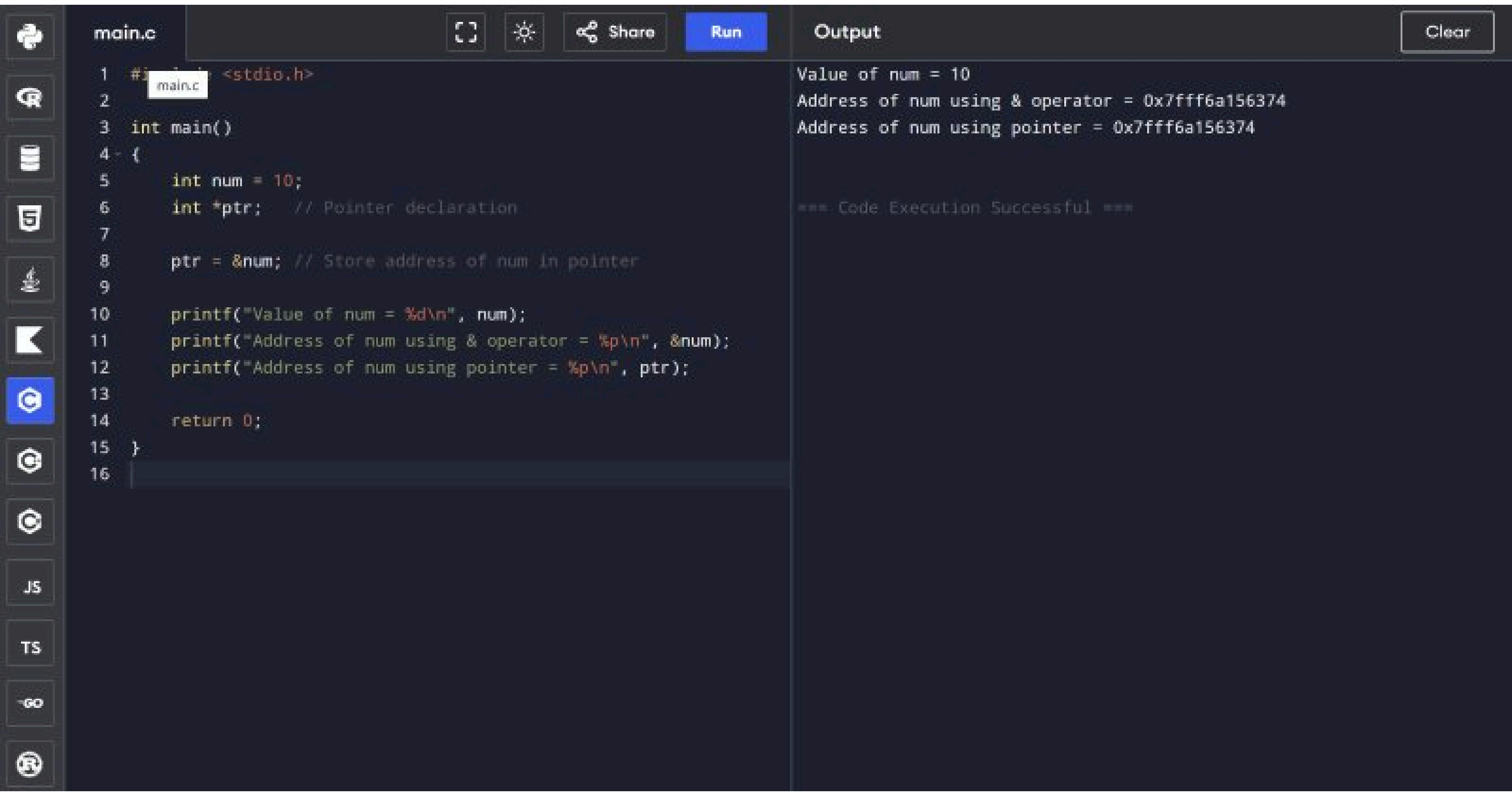
```c
#include <stdio.h>

// Function to swap numbers using pointers
void swap(int *a, int *b) {
    int temp;
    temp = *a;   // store value of a
    *a = *b;     // assign value of b to a
    *b = temp;   // assign temp to b
}

int main() {
    int num1, num2;

    printf("Enter two numbers:\n");
    scanf("%d %d", &num1, &num2);

    printf("Before swapping: num1 = %d, num2 = %d\n", num1,
        num2);

    // Pass addresses of num1 and num2
    swap(&num1, &num2);

    printf("After swapping: num1 = %d, num2 = %d\n", num1,
        num2);

    return 0;
}
```

```
Enter two numbers:
3 5
Before swapping: num1 = 3, num2 = 5
After swapping: num1 = 5, num2 = 3


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



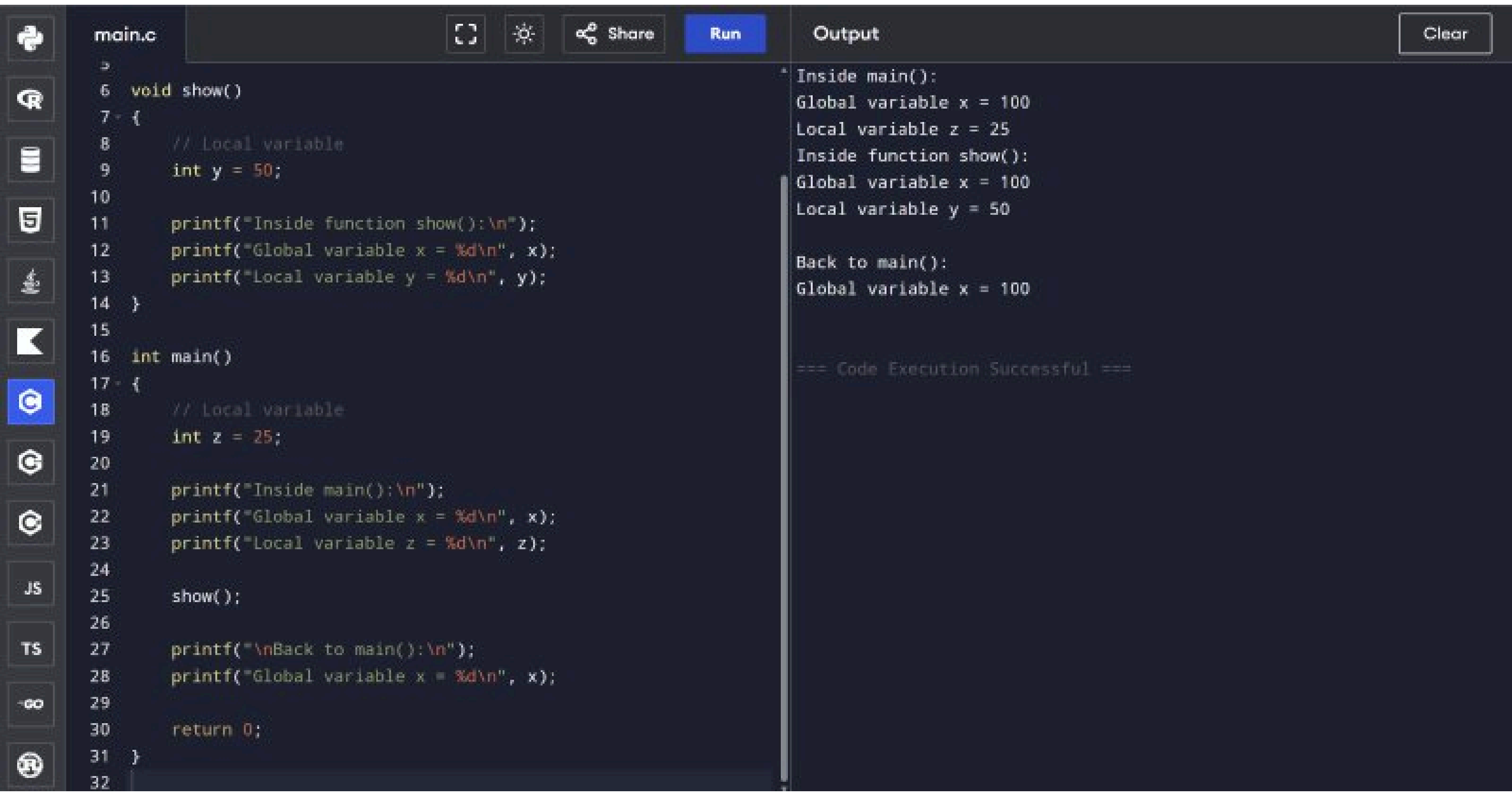
```c
#include <stdio.h>

int main() {
    int arr[5] = {10, 20, 30, 40, 50};
    int *ptr;
    int i;

    // Assign pointer to the base address of array
    ptr = arr;

    printf("Accessing array elements using pointers:\n");

    for(i = 0; i < 5; i++) {
        // Access elements using pointer
        printf("Element %d = %d\n", i, *(ptr + i));
    }

    return 0;
}
```

```
Accessing array elements using pointers:
Element 0 = 10
Element 1 = 20
Element 2 = 30
Element 3 = 40
Element 4 = 50


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



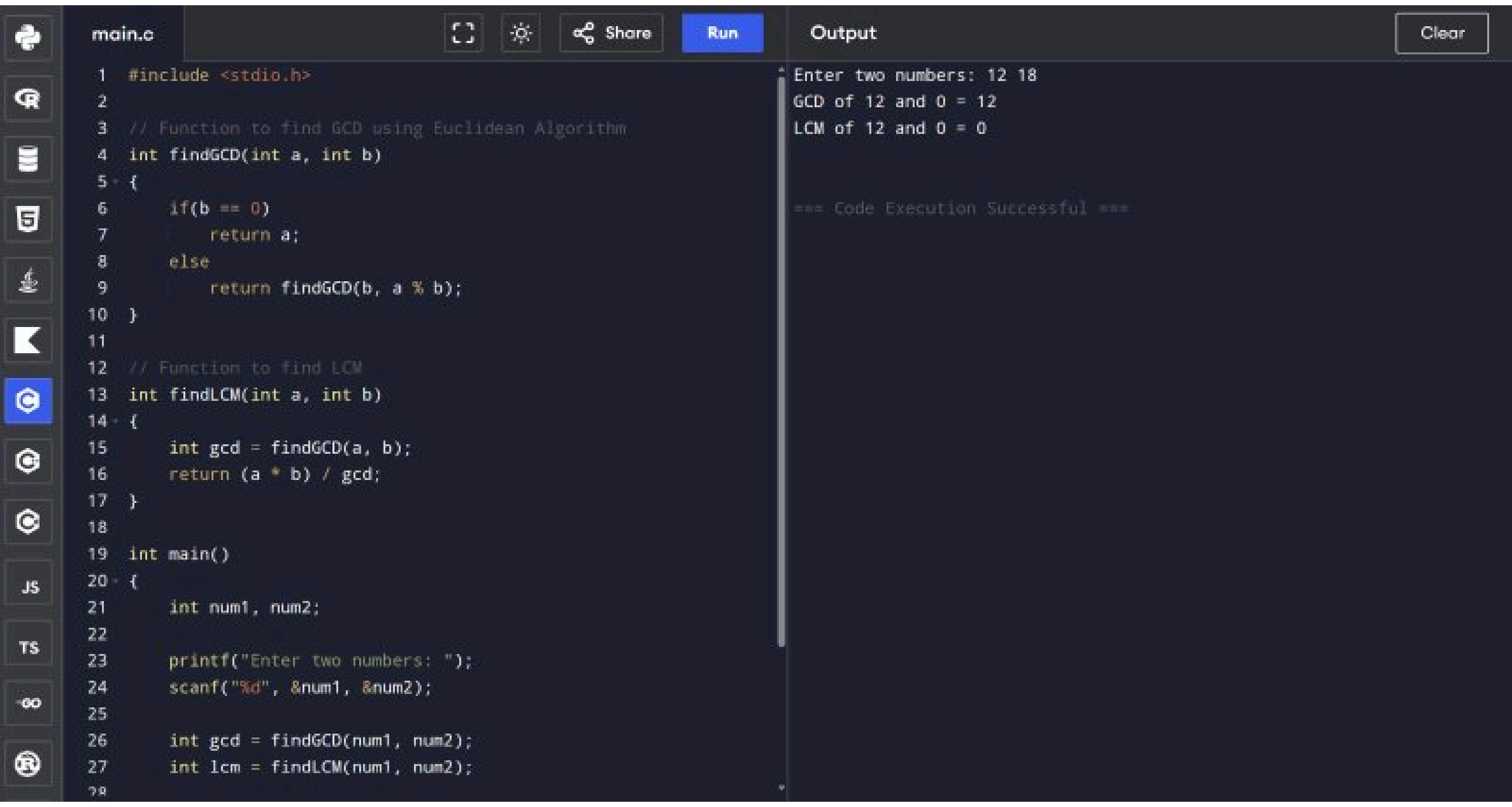
```c
#include <stdio.h>

int main()
{
    int num = 10;
    int *ptr;   // Pointer declaration

    ptr = &num; // Store address of num in pointer

    printf("Value of num = %d\n", num);
    printf("Address of num using & operator = %p\n", &num);
    printf("Address of num using pointer = %p\n", ptr);

    return 0;
}

```

```
Value of num = 10
Address of num using & operator = 0x7fff6a156374
Address of num using pointer = 0x7fff6a156374


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



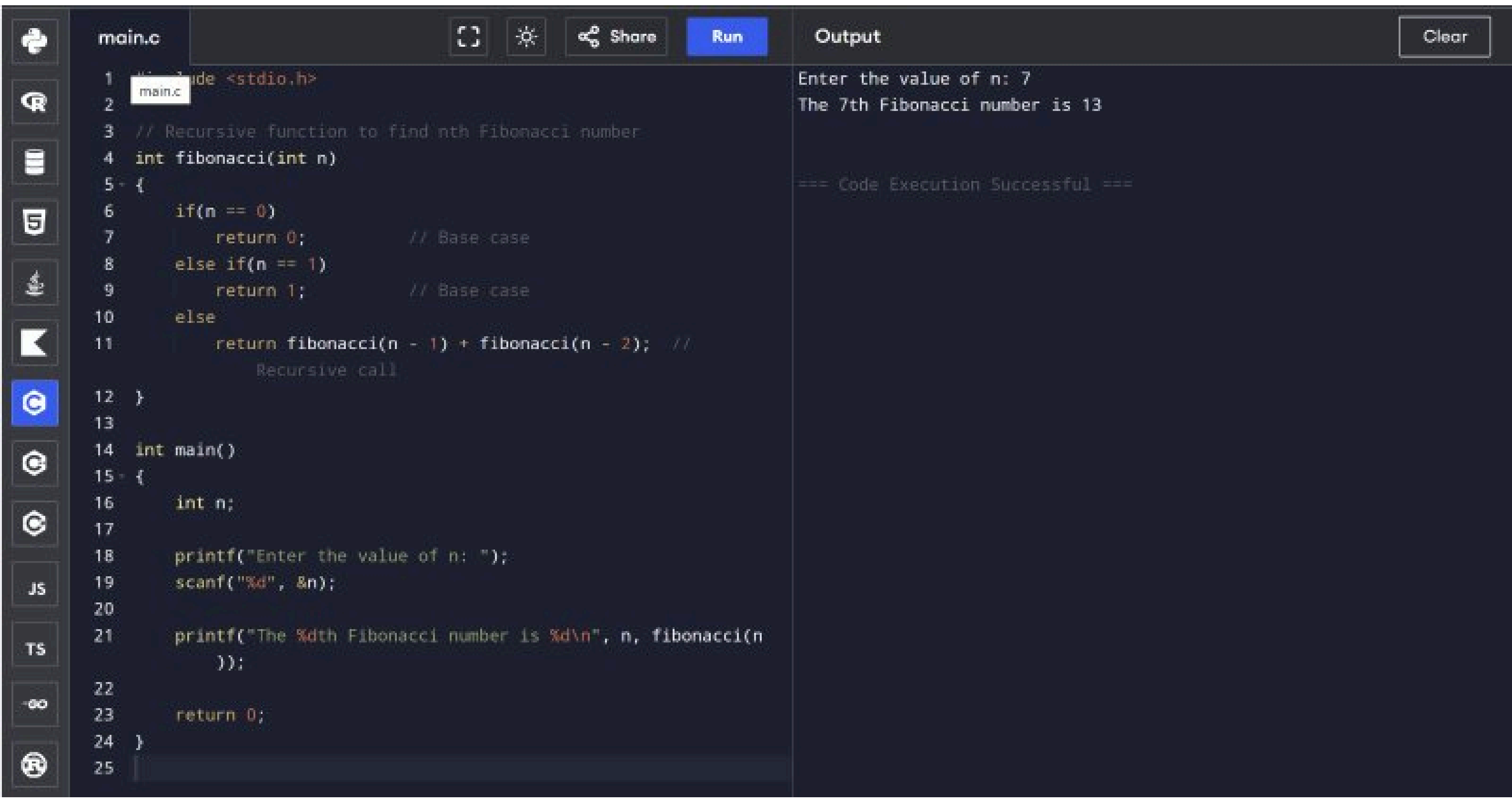
```c
void show()
{
    // Local variable
    int y = 50;

    printf("Inside function show():\n");
    printf("Global variable x = %d\n", x);
    printf("Local variable y = %d\n", y);
}

int main()
{
    // Local variable
    int z = 25;

    printf("Inside main():\n");
    printf("Global variable x = %d\n", x);
    printf("Local variable z = %d\n", z);

    show();

    printf("\nBack to main():\n");
    printf("Global variable x = %d\n", x);

    return 0;
}
```

```
Inside main():
Global variable x = 100
Local variable z = 25
Inside function show():
Global variable x = 100
Local variable y = 50

Back to main():
Global variable x = 100


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



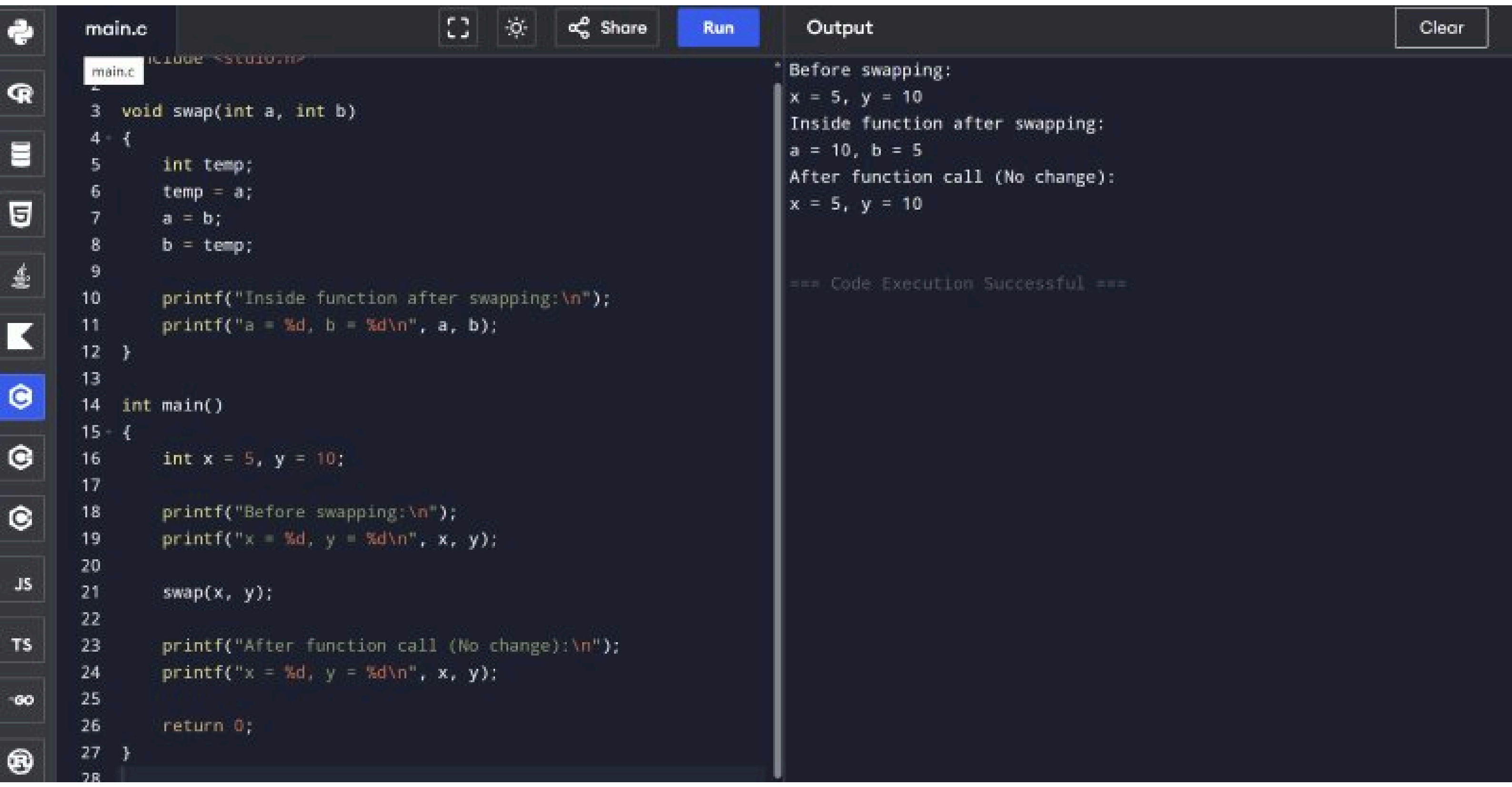
```c
#include <stdio.h>

// Function to find GCD using Euclidean Algorithm
int findGCD(int a, int b)
{
    if(b == 0)
        return a;
    else
        return findGCD(b, a % b);
}

// Function to find LCM
int findLCM(int a, int b)
{
    int gcd = findGCD(a, b);
    return (a * b) / gcd;
}

int main()
{
    int num1, num2;

    printf("Enter two numbers: ");
    scanf("%d", &num1, &num2);

    int gcd = findGCD(num1, num2);
    int lcm = findLCM(num1, num2);
```

```
Enter two numbers: 12 18
GCD of 12 and 0 = 12
LCM of 12 and 0 = 0


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



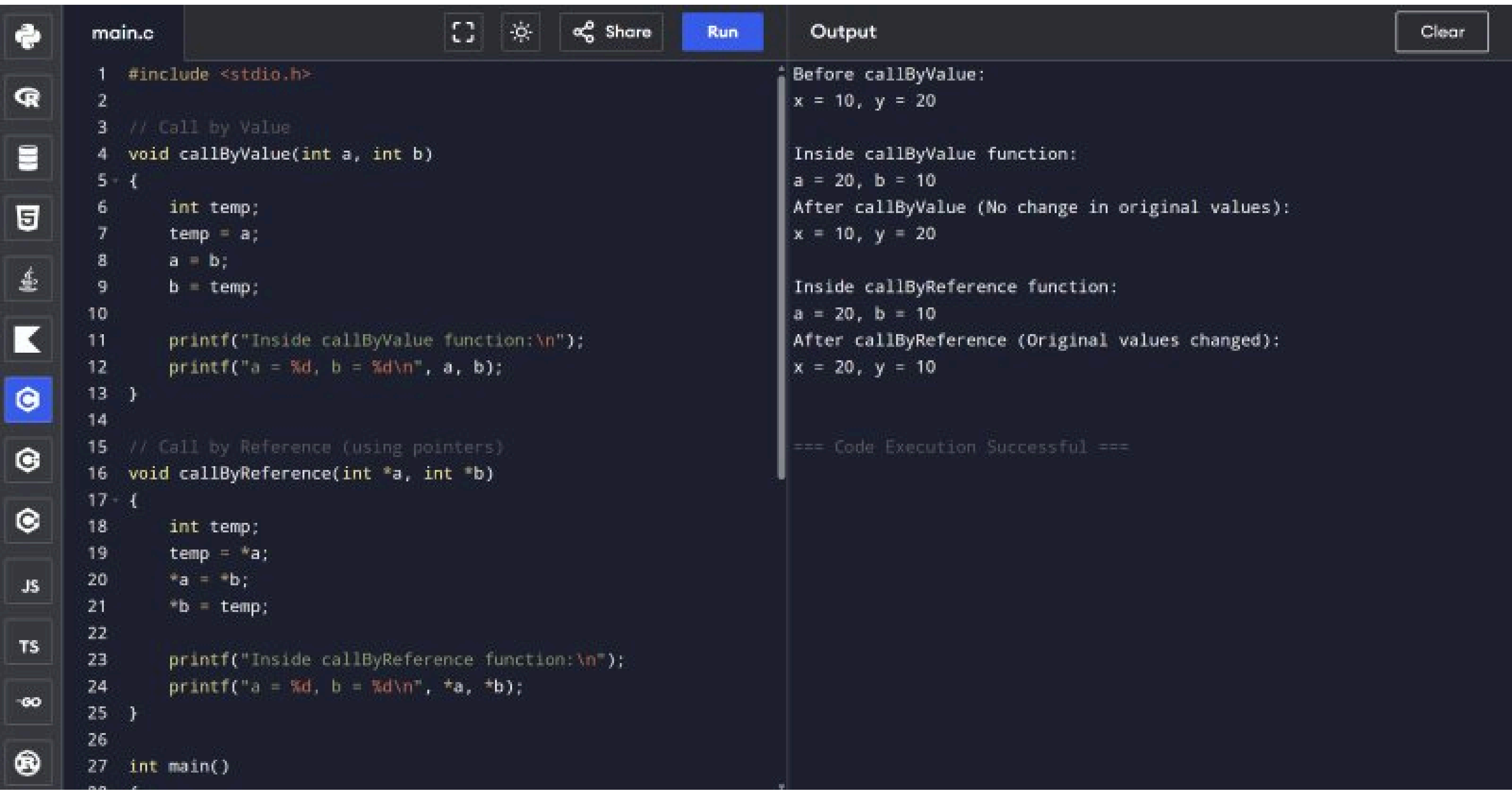
```c
#include <stdio.h>

// Recursive function to find nth Fibonacci number
int fibonacci(int n)
{
    if(n == 0)
        return 0;          // Base case
    else if(n == 1)
        return 1;          // Base case
    else
        return fibonacci(n - 1) + fibonacci(n - 2);  // Recursive call
}

int main()
{
    int n;

    printf("Enter the value of n: ");
    scanf("%d", &n);

    printf("The %dth Fibonacci number is %d\n", n, fibonacci(n));

    return 0;
}
```

```
Enter the value of n: 7
The 7th Fibonacci number is 13


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



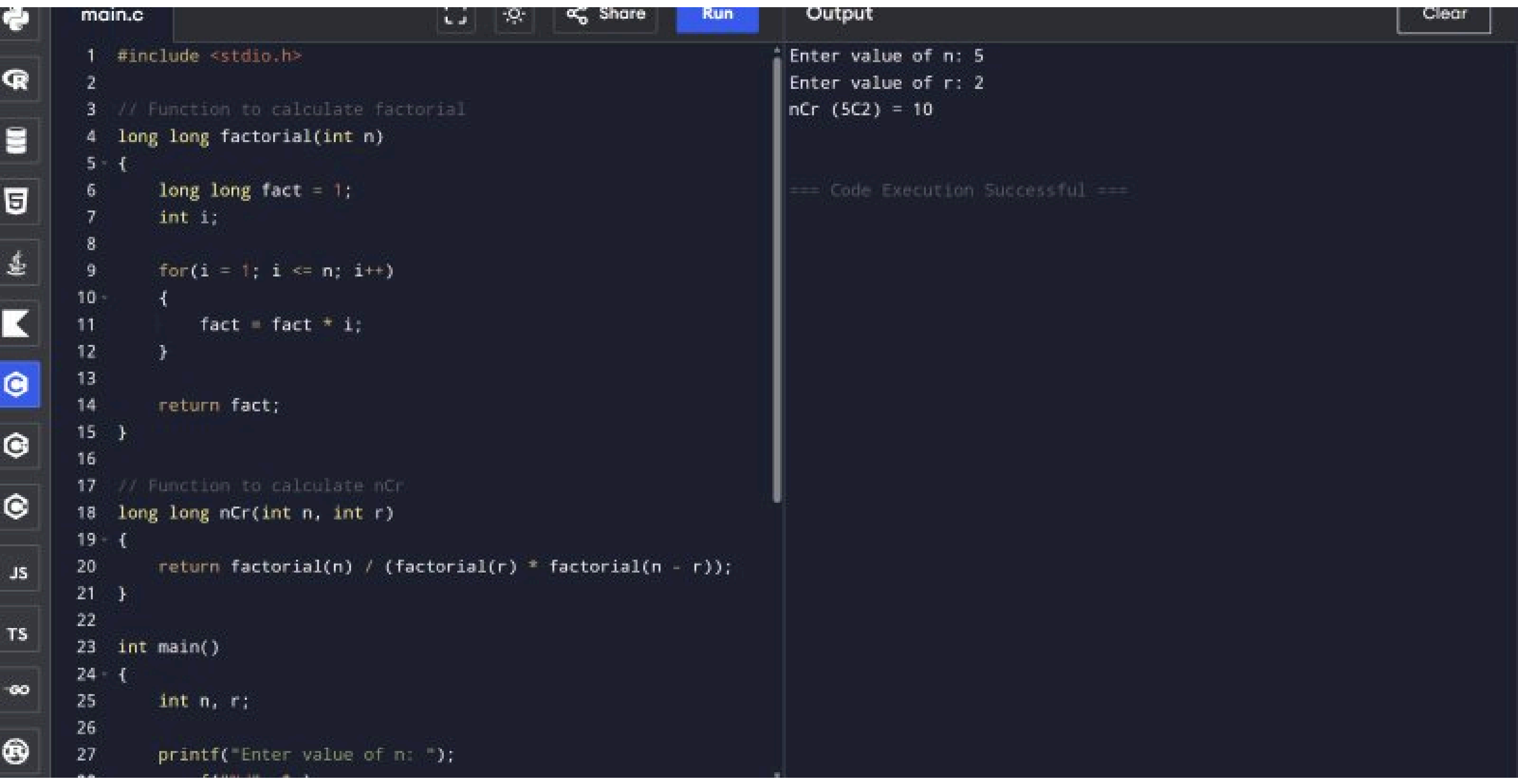
```c
nclude <stdio.h>

void swap(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;

    printf("Inside function after swapping:\n");
    printf("a = %d, b = %d\n", a, b);
}

int main()
{
    int x = 5, y = 10;

    printf("Before swapping:\n");
    printf("x = %d, y = %d\n", x, y);

    swap(x, y);

    printf("After function call (No change):\n");
    printf("x = %d, y = %d\n", x, y);

    return 0;
}
```

```
Before swapping:
x = 5, y = 10
Inside function after swapping:
a = 10, b = 5
After function call (No change):
x = 5, y = 10


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



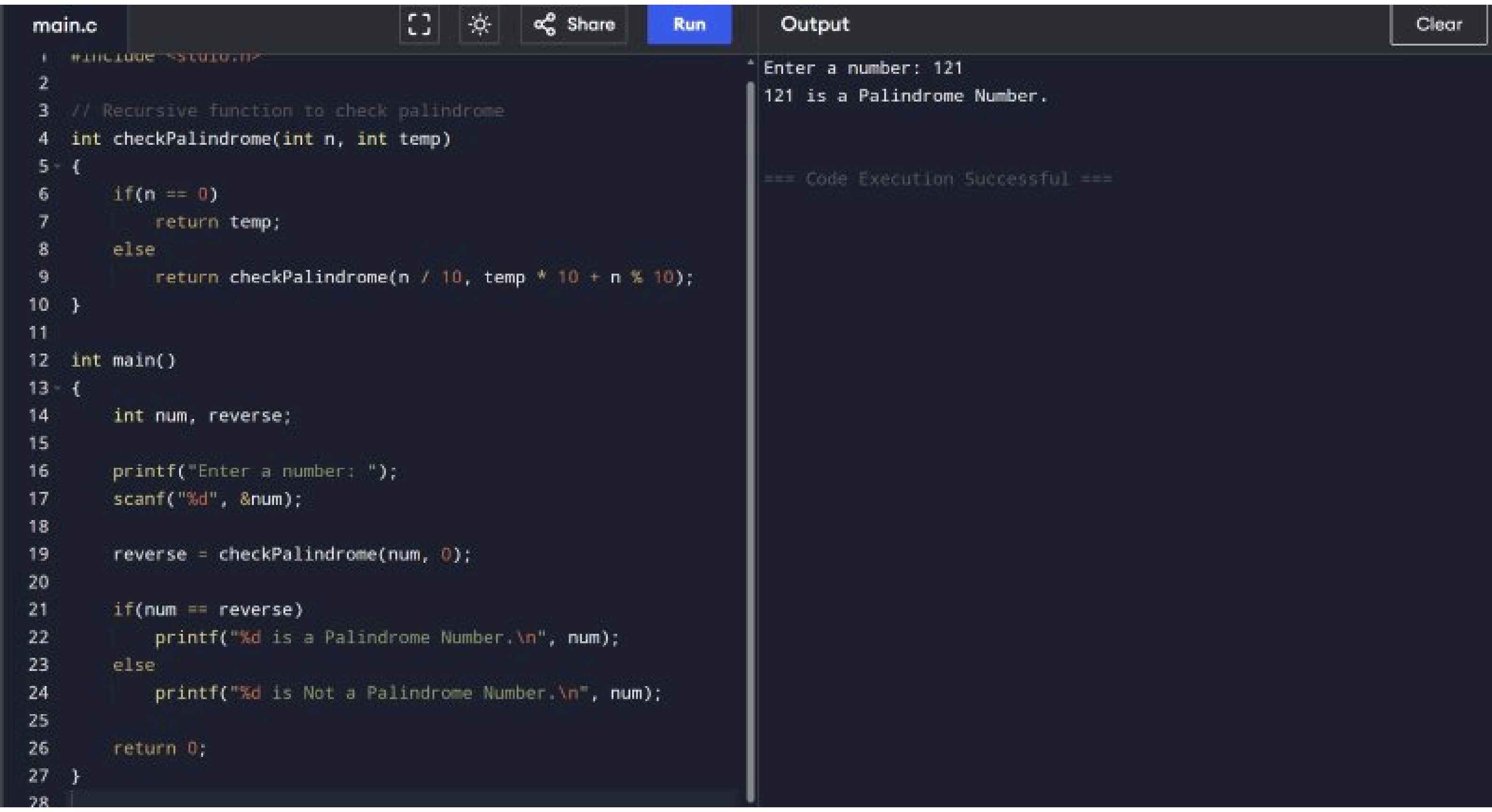
```c
#include <stdio.h>

// Call by Value
void callByValue(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;

    printf("Inside callByValue function:\n");
    printf("a = %d, b = %d\n", a, b);
}

// Call by Reference (using pointers)
void callByReference(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;

    printf("Inside callByReference function:\n");
    printf("a = %d, b = %d\n", *a, *b);
}

int main()
```

```
Before callByValue:
x = 10, y = 20

Inside callByValue function:
a = 20, b = 10
After callByValue (No change in original values):
x = 10, y = 20

Inside callByReference function:
a = 20, b = 10
After callByReference (Original values changed):
x = 20, y = 10


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



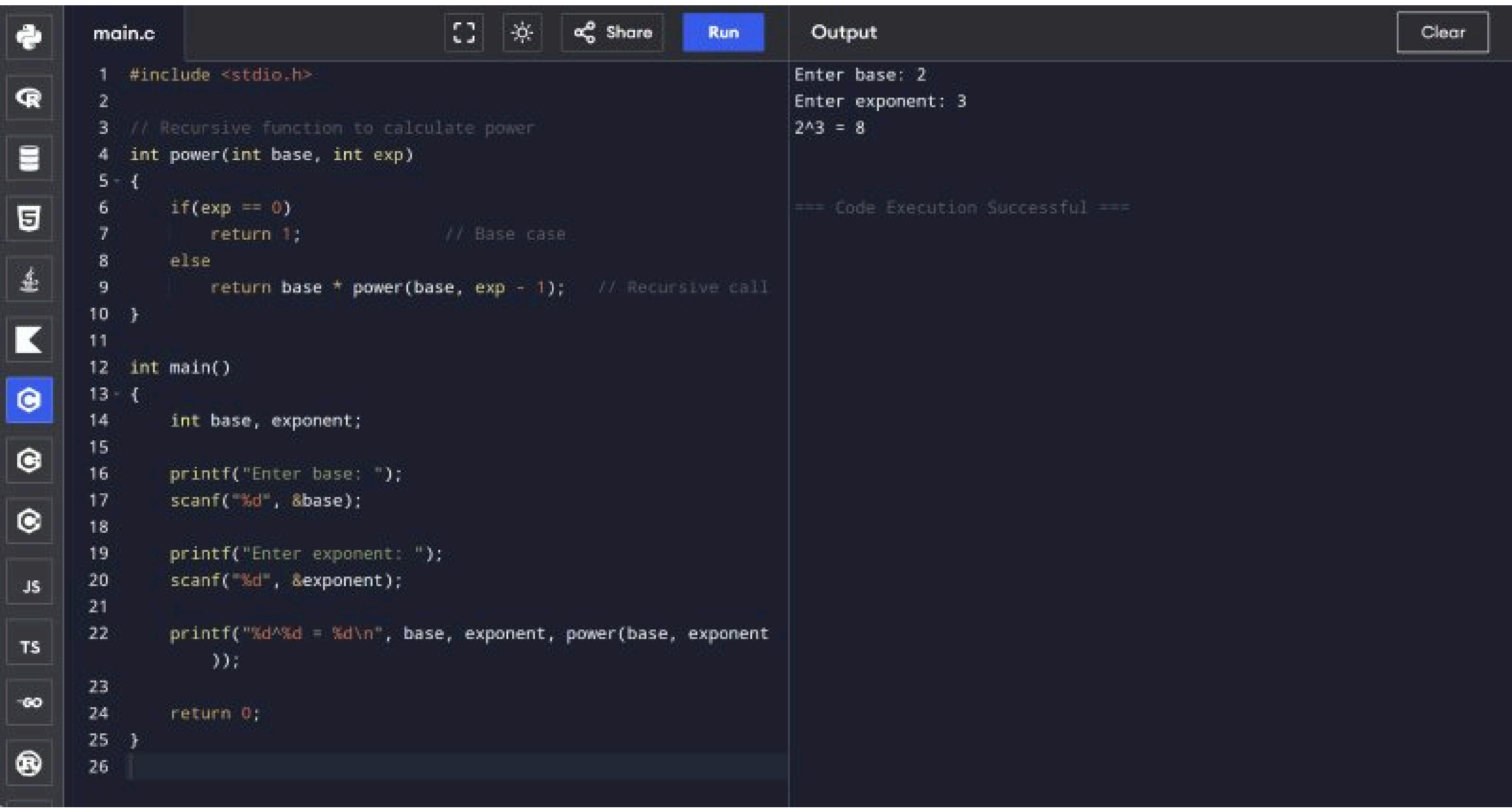
```c
#include <stdio.h>

// Function to calculate factorial
long long factorial(int n)
{
    long long fact = 1;
    int i;

    for(i = 1; i <= n; i++)
    {
        fact = fact * i;
    }

    return fact;
}

// Function to calculate nCr
long long nCr(int n, int r)
{
    return factorial(n) / (factorial(r) * factorial(n - r));
}

int main()
{
    int n, r;

    printf("Enter value of n: ");
```

```
Enter value of n: 5
Enter value of r: 2
nCr (5C2) = 10


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



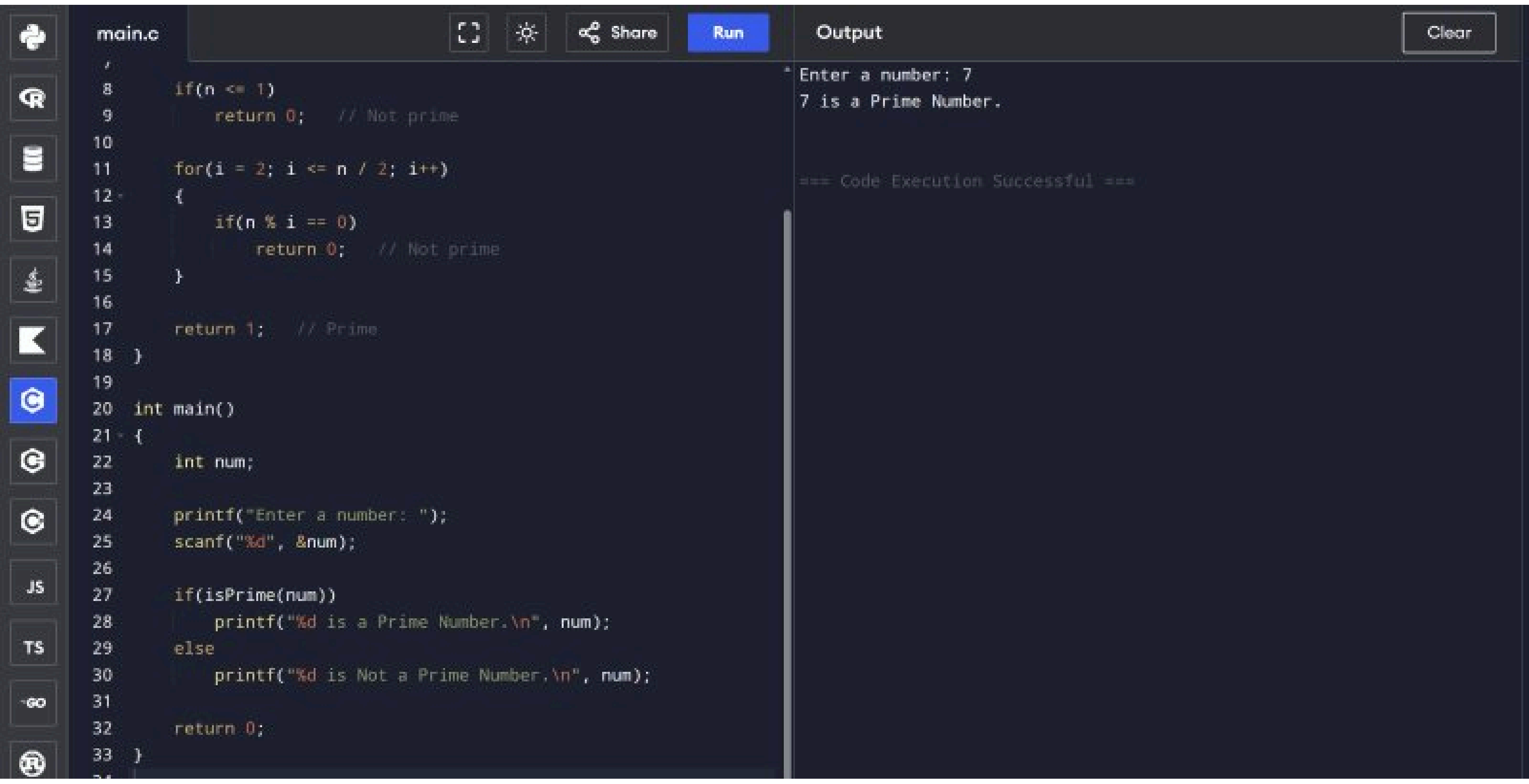
```c
#include <stdio.h>

// Recursive function to check palindrome
int checkPalindrome(int n, int temp)
{
    if(n == 0)
        return temp;
    else
        return checkPalindrome(n / 10, temp * 10 + n % 10);
}

int main()
{
    int num, reverse;

    printf("Enter a number: ");
    scanf("%d", &num);

    reverse = checkPalindrome(num, 0);

    if(num == reverse)
        printf("%d is a Palindrome Number.\n", num);
    else
        printf("%d is Not a Palindrome Number.\n", num);

    return 0;
}
```

```
Enter a number: 121
121 is a Palindrome Number.


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



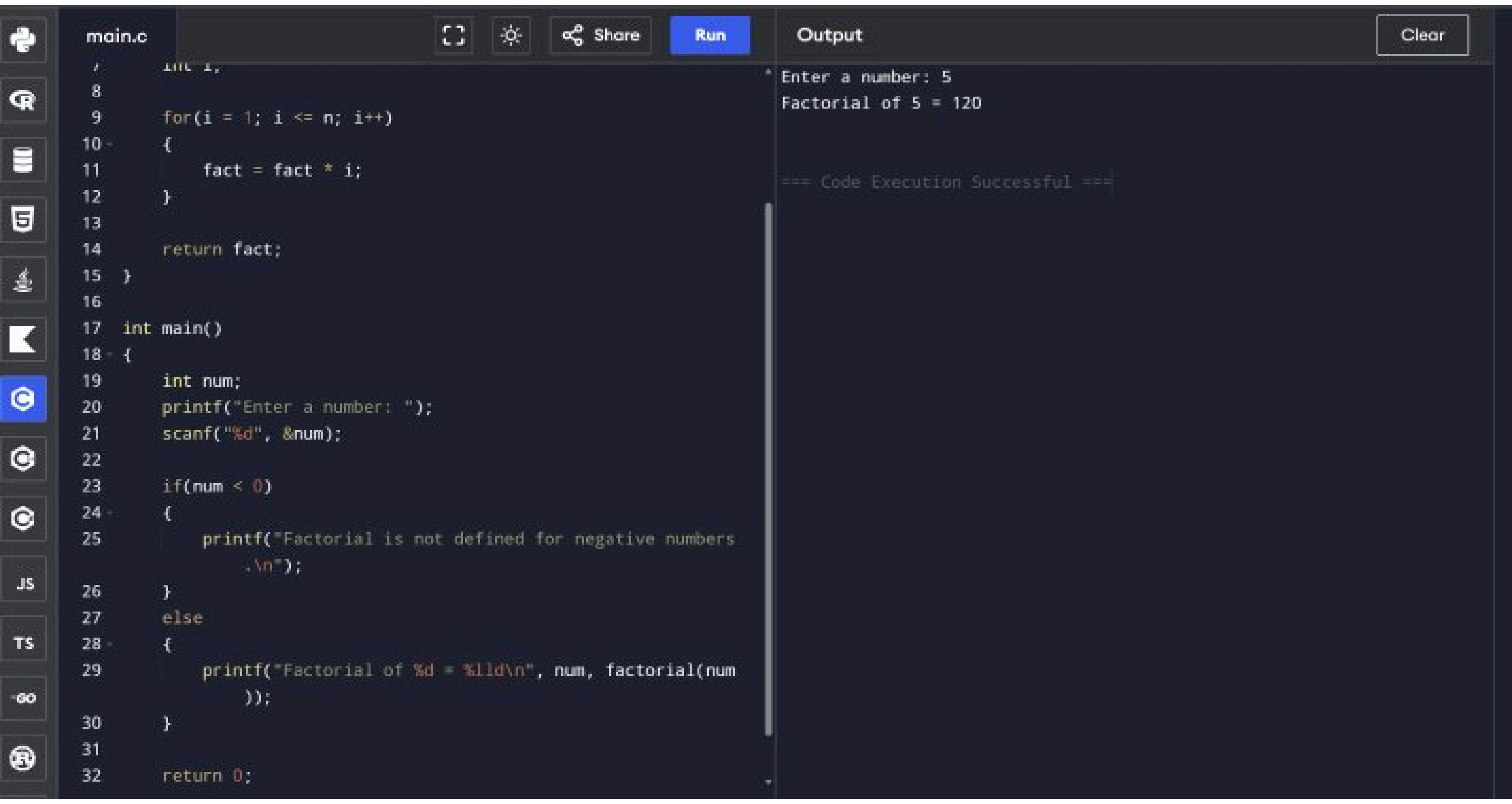
```c
#include <stdio.h>

// Recursive function to calculate power
int power(int base, int exp)
{
    if(exp == 0)
        return 1;              // Base case
    else
        return base * power(base, exp - 1);   // Recursive call
}

int main()
{
    int base, exponent;

    printf("Enter base: ");
    scanf("%d", &base);

    printf("Enter exponent: ");
    scanf("%d", &exponent);

    printf("%d^%d = %d\n", base, exponent, power(base, exponent
        ));

    return 0;
}
```

```
Enter base: 2
Enter exponent: 3
2^3 = 8


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



```c
    if(n <= 1)
        return 0;   // Not prime

    for(i = 2; i <= n / 2; i++)
    {
        if(n % i == 0)
            return 0;   // Not prime
    }

    return 1;   // Prime
}

int main()
{
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    if(isPrime(num))
        printf("%d is a Prime Number.\n", num);
    else
        printf("%d is Not a Prime Number.\n", num);

    return 0;
}
```

```
Enter a number: 7
7 is a Prime Number.


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



```c
    int i;

    for(i = 1; i <= n; i++)
    {
        fact = fact * i;
    }

    return fact;
}

int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    if(num < 0)
    {
        printf("Factorial is not defined for negative numbers
            .\n");
    }
    else
    {
        printf("Factorial of %d = %lld\n", num, factorial(num
            ));
    }

    return 0;
```

```
Enter a number: 5
Factorial of 5 = 120


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