

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various languages and tools. The main area displays a C program named `main.c`. The code defines a factorial function and calls it from the main function, handling negative numbers by printing an error message. The output window shows the execution results, including the input prompt, the calculated factorial value, and a success message.

```
main.c
1  #include <stdio.h>
2
3  int factorial(int n)
4  {
5      int fact = 1;
6      for(i = 1; i <= n; i++)
7      {
8          fact = fact * i;
9      }
10     return fact;
11 }
12
13
14 int main()
15 {
16     int num;
17     printf("Enter a number: ");
18     scanf("%d", &num);
19
20     if(num < 0)
21     {
22         printf("Factorial is not defined for negative numbers
23             .\n");
24     }
25     else
26     {
27         printf("Factorial of %d = %lld\n", num, factorial(num
28             ));
29     }
30 }
31
32 return 0;
```

Output

```
* Enter a number: 5
Factorial of 5 = 120
==== Code Execution Successful ====
```

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various languages and tools. The main area displays a C program named 'main.c'. The code allocates memory for an array of integers, reads two elements from the user, prints them, and then frees the allocated memory.

```
main.c
12     ptr = (int *)malloc(n * sizeof(int));
13
14     // Check if memory is allocated successfully
15     if (ptr == NULL) {
16         printf("Memory allocation failed!\n");
17         return 1;
18     }
19
20     // Input elements
21     printf("Enter %d elements:\n", n);
22     for (i = 0; i < n; i++) {
23         scanf("%d", (ptr + i));
24     }
25
26     // Display elements
27     printf("Entered elements are:\n");
28     for (i = 0; i < n; i++) {
29         printf("%d ", *(ptr + i));
30     }
31
32     // Free the allocated memory
33     free(ptr);
34
35     printf("\nMemory successfully freed.\n");
36
37     return 0;
38 }
```

The 'Run' button is highlighted in blue at the top right. To its right is an 'Output' panel which shows the execution results:

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

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

The screenshot shows a code editor interface with a sidebar containing various language icons (Python, C, C++, JavaScript, TypeScript, Go, and others). The main area has tabs for "main.c" and "Output".

Code (main.c):

```
1 #include <stdio.h>
2
3 int main() {
4     int num = 10;
5     int *ptr;      // Pointer to int
6     int **dptr;    // Pointer to pointer
7
8     ptr = &num;    // ptr stores address of num
9     dptr = &ptr;   // dptr stores address of ptr
10
11    printf("Value of num = %d\n", num);
12    printf("Address of num = %p\n", &num);
13
14    printf("Value of ptr (address of num) = %p\n", ptr);
15    printf("Address of ptr = %p\n", &ptr);
16
17    printf("Value of dptr (address of ptr) = %p\n", dptr);
18
19    // Accessing value using pointer to pointer
20    printf("Value using *ptr = %d\n", *ptr);
21    printf("Value using **dptr = %d\n", **dptr);
22
23    return 0;
24 }
25 }
```

Output:

```
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 ===
```

main.o

main.c



Run

Clear

Output

```
* Enter a string: saveetha
Number of vowels = 4
*** Code Execution Successful ***
```

```
3 int main() {
4     char str[100];
5     char *ptr;
6     int count = 0;
7
8     printf("Enter a string: ");
9     fgets(str, sizeof(str), stdin);
10
11    // Point to the first character
12    ptr = str;
13
14    // Traverse string using pointer
15    while (*ptr != '\0') {
16        if (*ptr == 'a' || *ptr == 'e' || *ptr == 'i' || *ptr
17            == 'o' || *ptr == 'u' ||
18            *ptr == 'A' || *ptr == 'E' || *ptr == 'I' || *ptr
19            == 'O' || *ptr == 'U') {
20            count++;
21        }
22        ptr++; // Move pointer to next character
23    }
24
25    printf("Number of vowels = %d\n", count);
26
27 }
```

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various languages and tools. The main area displays a C program named `main.c`. The code reads a string from standard input, finds its end, and then reverses it using pointer swapping. The output window shows the input "hello" and the reversed output "olleh", followed by a success message.

```
main.c
1  printf("Enter a string: ");
2  fgets(str, sizeof(str), stdin);
3
4  // Set start pointer to beginning of string
5  start = str;
6
7  // Find the end of the string
8  end = str;
9  while (*end != '\0' && *end != '\n') {
10     end++;
11 }
12 end--; // Move back to last character (before '\0')
13
14 // Reverse the string using pointers
15 while (start < end) {
16     temp = *start;
17     *start = *end;
18     *end = temp;
19
20     start++;
21     end--;
22 }
23
24 printf("Reversed string: %s", str);
25
26 return 0;
27 }
```

Output

Enter a string: hello
Reversed string: olleh
==== Code Execution Successful ===

main.c



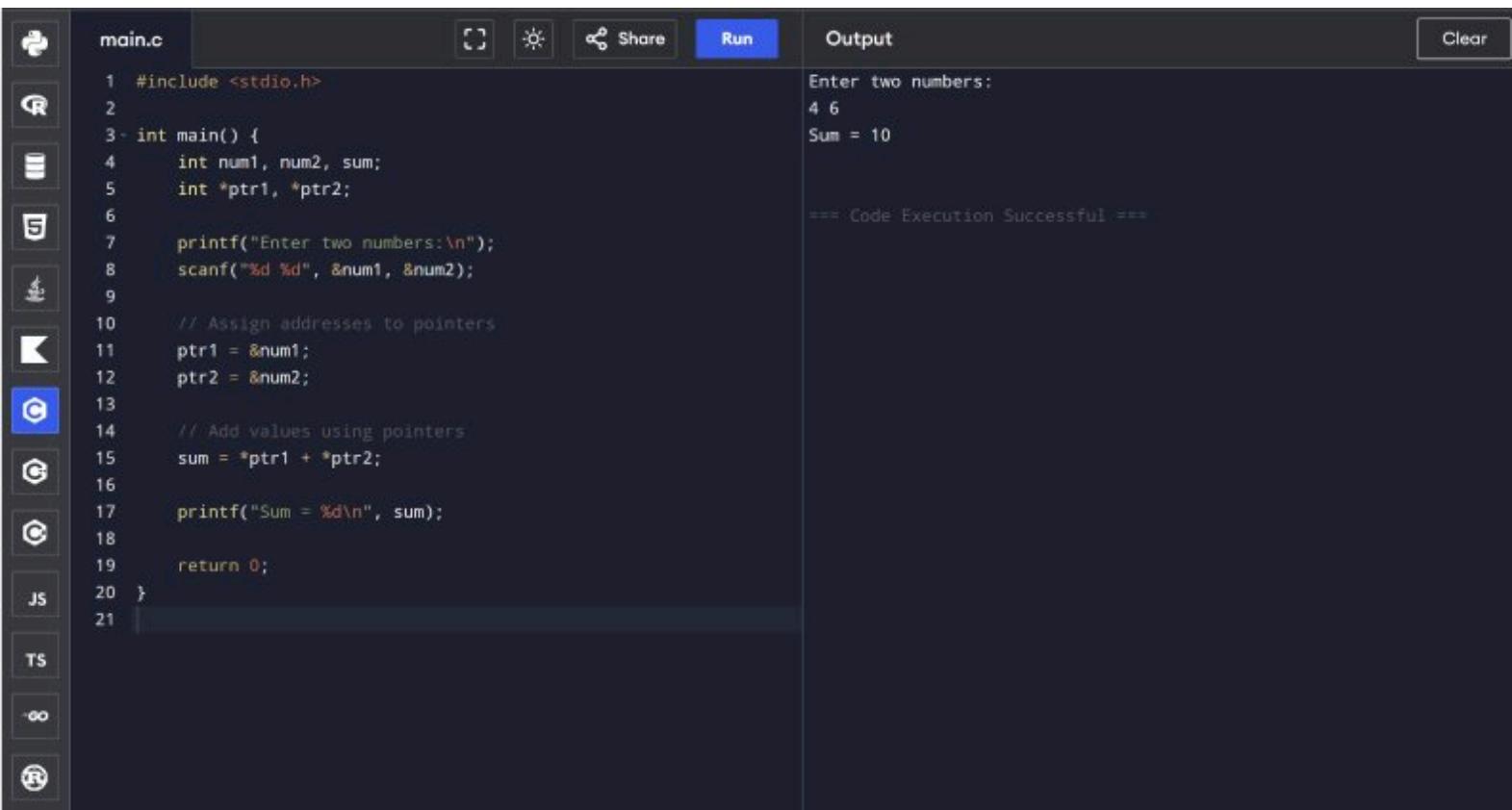
Run

Output

Clear

```
1 #include <stdio.h>
2
3 int main() {
4     char str[100];
5     char *ptr;
6     int length = 0;
7
8     printf("Enter a string: ");
9     fgets(str, sizeof(str), stdin);
10
11    // Point to the first character of the string
12    ptr = str;
13
14    // Traverse string using pointer
15    while (*ptr != '\0' && *ptr != '\n') {
16        length++;
17        ptr++; // Move pointer to next character
18    }
19
20    printf("Length of the string = %d\n", length);
21
22    return 0;
23 }
```

```
Enter a string: hello
Length of the string = 5
== Code Execution Successful ==
```



main.c

```
1 #include <stdio.h>
2
3 int main() {
4     int num1, num2, sum;
5     int *ptr1, *ptr2;
6
7     printf("Enter two numbers:\n");
8     scanf("%d %d", &num1, &num2);
9
10    // Assign addresses to pointers
11    ptr1 = &num1;
12    ptr2 = &num2;
13
14    // Add values using pointers
15    sum = *ptr1 + *ptr2;
16
17    printf("Sum = %d\n", sum);
18
19    return 0;
20 }
21
```



Run

Output

Clear

```
Enter two numbers:
4 6
Sum = 10
===[Code Execution Successful]==
```

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with various icons for different file types and operations. The main area has tabs for "main.c" and "Output". The "main.c" tab is active, displaying the following C code:

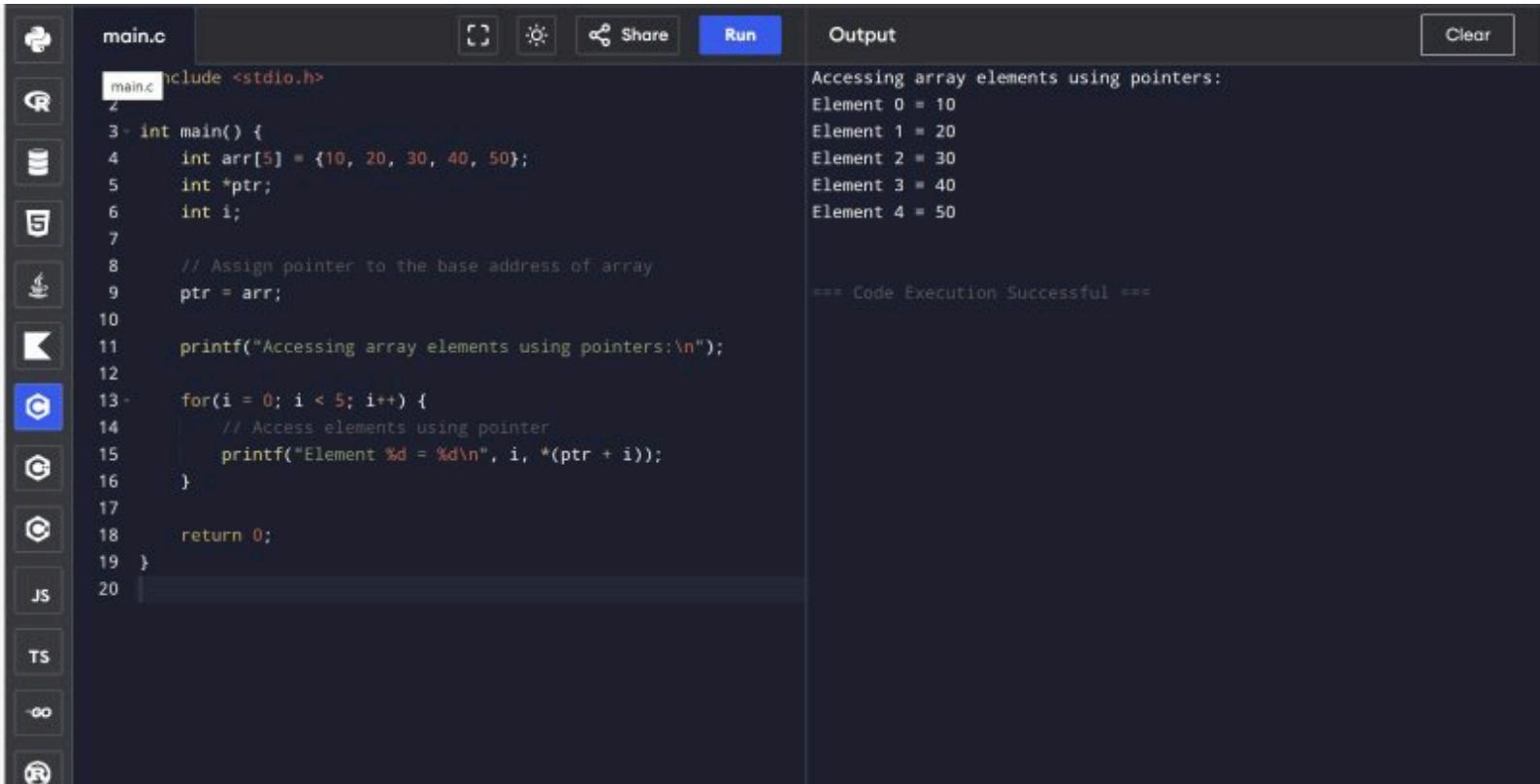
```
1 #include <stdio.h>
2
3 // Function to swap numbers using pointers
4 void swap(int *a, int *b) {
5     int temp;
6     temp = *a; // store value of a
7     *a = *b;    // assign value of b to a
8     *b = temp; // assign temp to b
9 }
10
11 int main() {
12     int num1, num2;
13
14     printf("Enter two numbers:\n");
15     scanf("%d %d", &num1, &num2);
16
17     printf("Before swapping: num1 = %d, num2 = %d\n", num1,
18           num2);
19
20     // Pass addresses of num1 and num2
21     swap(&num1, &num2);
22
23     printf("After swapping: num1 = %d, num2 = %d\n", num1,
24           num2);
25 }
26
```

The "Output" tab is also active, showing the execution results:

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

*** Code Execution Successful ***
```

A "Run" button is located at the top of the editor, just below the tabs.



main.c

```
1 main.c
2
3 int main() {
4     int arr[5] = {10, 20, 30, 40, 50};
5     int *ptr;
6     int i;
7
8     // Assign pointer to the base address of array
9     ptr = arr;
10
11    printf("Accessing array elements using pointers:\n");
12
13    for(i = 0; i < 5; i++) {
14        // Access elements using pointer
15        printf("Element %d = %d\n", i, *(ptr + i));
16    }
17
18    return 0;
19 }
```

Run

Output

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

== Code Execution Successful ==

Clear



The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various languages and tools. The main area has tabs for 'main.c' and 'main.o'. The code editor contains the following C code:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int num = 10;
6     int *ptr; // Pointer declaration
7
8     ptr = &num; // Store address of num in pointer
9
10    printf("Value of num = %d\n", num);
11    printf("Address of num using & operator = %p\n", &num);
12    printf("Address of num using pointer = %p\n", ptr);
13
14    return 0;
15 }
16
```

At the top of the editor, there are several buttons: a file icon, a copy icon, a paste icon, a share icon, a run button, and a clear button.

The 'Output' panel on the right displays the results of the program execution:

```
Value of num = 10
Address of num using & operator = 0x7fff6a156374
Address of num using pointer = 0x7fff6a156374
*** Code Execution Successful ***
```

The screenshot shows a code editor interface with a dark theme. On the left, there's a vertical toolbar with icons for various languages and tools. The main area displays a C program named `main.c`. The code defines a global variable `x` (line 5), a local variable `y` in `show()` (line 9), and a local variable `z` in `main()` (line 19). It also includes `printf` statements to print the values of these variables.

```
main.c
6 void show()
7 {
8     // Local variable
9     int y = 50;
10
11    printf("Inside function show():\n");
12    printf("Global variable x = %d\n", x);
13    printf("Local variable y = %d\n", y);
14 }
15
16 int main()
17 {
18     // Local variable
19     int z = 25;
20
21     printf("Inside main():\n");
22     printf("Global variable x = %d\n", x);
23     printf("Local variable z = %d\n", z);
24
25     show();
26
27     printf("\nBack to main():\n");
28     printf("Global variable x = %d\n", x);
29
30     return 0;
31 }
```

The `Run` button is highlighted in blue at the top right. To its right is a `Clear` button. The `Output` panel on the right shows the execution results:

```
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 ---
```

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various languages and tools. The main area has tabs for "main.c" and "Output". The "main.c" tab contains the following C code:

```
1 #include <stdio.h>
2
3 // Function to find GCD using Euclidean Algorithm
4 int findGCD(int a, int b)
5 {
6     if(b == 0)
7         return a;
8     else
9         return findGCD(b, a % b);
10 }
11
12 // Function to find LCM
13 int findLCM(int a, int b)
14 {
15     int gcd = findGCD(a, b);
16     return (a * b) / gcd;
17 }
18
19 int main()
20 {
21     int num1, num2;
22
23     printf("Enter two numbers: ");
24     scanf("%d", &num1, &num2);
25
26     int gcd = findGCD(num1, num2);
27     int lcm = findLCM(num1, num2);
28 }
```

The "Output" tab shows the execution results:

```
Enter two numbers: 12 18
GCD of 12 and 0 = 12
LCM of 12 and 0 = 0
*** Code Execution Successful ***
```

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various file types and settings. The main area has tabs for "main.c" and "Output".

Code (main.c):

```
1 #include <stdio.h>
2
3 // Recursive function to find nth Fibonacci number
4 int fibonacci(int n)
5 {
6     if(n == 0)
7         return 0;          // Base case
8     else if(n == 1)
9         return 1;          // Base case
10    else
11        return fibonacci(n - 1) + fibonacci(n - 2); // Recursive call
12 }
13
14 int main()
15 {
16     int n;
17
18     printf("Enter the value of n: ");
19     scanf("%d", &n);
20
21     printf("The %dth Fibonacci number is %d\n", n, fibonacci(n));
22
23     return 0;
24 }
```

Output:

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

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

The screenshot shows a code editor interface with a dark theme. On the left, there's a vertical toolbar with icons for various file types and tools. The main area has tabs for 'main.c' and 'INCLUDE <STUDIO.H>'. The code editor displays the following C code:

```
main.c
3 void swap(int a, int b)
4 {
5     int temp;
6     temp = a;
7     a = b;
8     b = temp;
9
10    printf("Inside function after swapping:\n");
11    printf("a = %d, b = %d\n", a, b);
12 }
13
14 int main()
15 {
16     int x = 5, y = 10;
17
18     printf("Before swapping:\n");
19     printf("x = %d, y = %d\n", x, y);
20
21     swap(x, y);
22
23     printf("After function call (No change):\n");
24     printf("x = %d, y = %d\n", x, y);
25
26     return 0;
27 }
28
```

The 'Run' button is highlighted in blue. To the right, the 'Output' tab is active, showing the execution results:

```
* 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 ***
```

main.c



Run

Output

Clear

```
1 #include <stdio.h>
2
3 // Call by Value
4 void callByValue(int a, int b)
5 {
6     int temp;
7     temp = a;
8     a = b;
9     b = temp;
10
11    printf("Inside callByValue function:\n");
12    printf("a = %d, b = %d\n", a, b);
13 }
14
15 // Call by Reference (using pointers)
16 void callByReference(int *a, int *b)
17 {
18     int temp;
19     temp = *a;
20     *a = *b;
21     *b = temp;
22
23    printf("Inside callByReference function:\n");
24    printf("a = %d, b = %d\n", *a, *b);
25 }
26
27 int main()
28 {
```

```
^ 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 ===
```

main.c

Run

Output

Clear

```
1 #include <stdio.h>
2
3 // Function to calculate factorial
4 long long factorial(int n)
5 {
6     long long fact = 1;
7     int i;
8
9     for(i = 1; i <= n; i++)
10    {
11        fact = fact * i;
12    }
13
14    return fact;
15 }
16
17 // Function to calculate nCr
18 long long nCr(int n, int r)
19 {
20     return factorial(n) / (factorial(r) * factorial(n - r));
21 }
22
23 int main()
24 {
25     int n, r;
26
27     printf("Enter value of n: ");
28     scanf("%d", &n);
29
30     printf("Enter value of r: ");
31     scanf("%d", &r);
32
33     long long result = nCr(n, r);
34
35     printf("nCr (%dC%d) = %lld", n, r, result);
36 }
```

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

--- Code Execution Successful ---
```

main.c

Run Clear

```
1 #include <stdio.h>
2
3 // Recursive function to check palindrome
4 int checkPalindrome(int n, int temp)
5 {
6     if(n == 0)
7         return temp;
8     else
9         return checkPalindrome(n / 10, temp * 10 + n % 10);
10 }
11
12 int main()
13 {
14     int num, reverse;
15
16     printf("Enter a number: ");
17     scanf("%d", &num);
18
19     reverse = checkPalindrome(num, 0);
20
21     if(num == reverse)
22         printf("%d is a Palindrome Number.\n", num);
23     else
24         printf("%d is Not a Palindrome Number.\n", num);
25
26     return 0;
27 }
28
```

Output

Enter a number: 121
121 is a Palindrome Number.
*** Code Execution Successful ***

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various languages and tools. The main area displays a C program named `main.c`. The code defines a recursive function `power` and a `main` function that reads user input for base and exponent, then prints the result of the power calculation.

```
main.c
1 #include <stdio.h>
2
3 // Recursive function to calculate power
4 int power(int base, int exp)
5 {
6     if(exp == 0)
7         return 1; // Base case
8     else
9         return base * power(base, exp - 1); // Recursive call
10 }
11
12 int main()
13 {
14     int base, exponent;
15
16     printf("Enter base: ");
17     scanf("%d", &base);
18
19     printf("Enter exponent: ");
20     scanf("%d", &exponent);
21
22     printf("%d^%d = %d\n", base, exponent, power(base, exponent));
23
24     return 0;
25 }
```

The output window shows the execution results:

```
Enter base: 2
Enter exponent: 3
2^3 = 8
==== Code Execution Successful ===
```

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various languages and tools. The main area displays a C program named `main.c`. The code implements a prime number checker. It defines a function `isPrime` that returns 1 if a number is prime and 0 if it is not. The main function prompts the user for a number, calls `isPrime`, and prints the result. The output window shows the execution of the code with the input 7, resulting in the output "7 is a Prime Number." followed by a success message.

```
main.c
8     if(n <= 1)
9         return 0; // Not prime
10
11    for(i = 2; i <= n / 2; i++)
12    {
13        if(n % i == 0)
14            return 0; // Not prime
15    }
16
17    return 1; // Prime
18 }
19
20 int main()
21 {
22     int num;
23
24     printf("Enter a number: ");
25     scanf("%d", &num);
26
27     if(isPrime(num))
28         printf("%d is a Prime Number.\n", num);
29     else
30         printf("%d is Not a Prime Number.\n", num);
31
32     return 0;
33 }
```

Output

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

*** Code Execution Successful ***
```

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for various languages and tools. The main area displays a C program named `main.c`. The code defines a factorial function and calls it from the main function, handling negative numbers by printing an error message. The output window shows the execution results, including the input prompt, the calculated factorial value, and a success message.

```
main.c
1  #include <stdio.h>
2
3  int factorial(int n)
4  {
5      int fact = 1;
6
7      for(i = 1; i <= n; i++)
8      {
9          fact = fact * i;
10     }
11
12     return fact;
13 }
14
15 }
16
17 int main()
18 {
19     int num;
20     printf("Enter a number: ");
21     scanf("%d", &num);
22
23     if(num < 0)
24     {
25         printf("Factorial is not defined for negative numbers
26             .\n");
27     }
28     else
29     {
30         printf("Factorial of %d = %lld\n", num, factorial(num
31             ));
32     }
33
34     return 0;
35 }
```

Output

```
* Enter a number: 5
Factorial of 5 = 120
==== Code Execution Successful ====
```