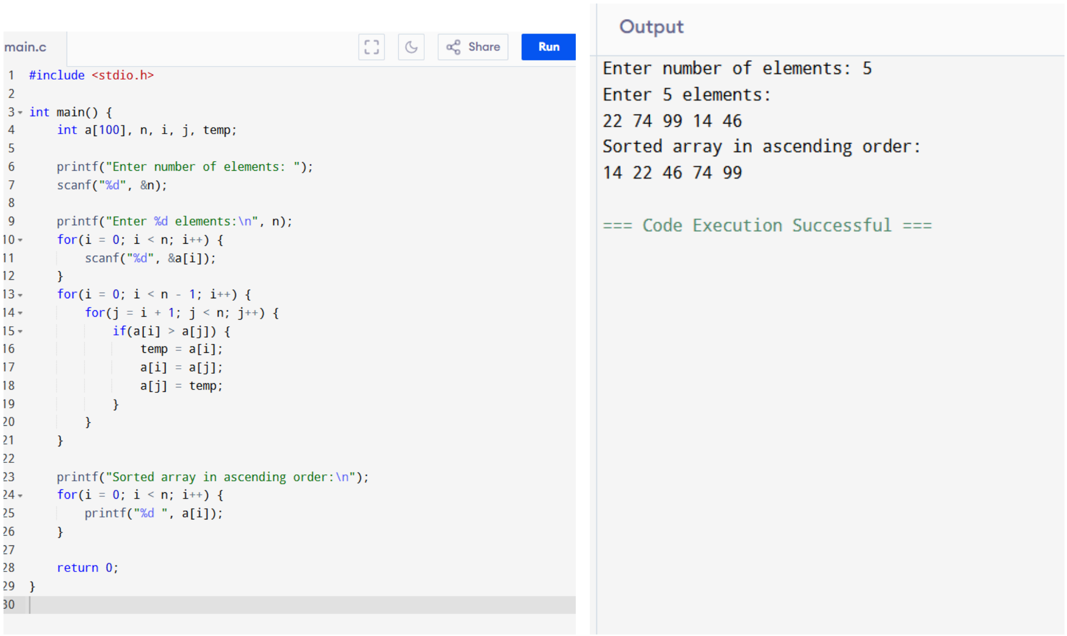
1. Selection Sort (Ascending Order)

Aim: To write a C program to sort an array of elements in ascending order using the Selection Sort algorithm.

Algorithm Steps:

```
1. Start
2. Read the number of elements, n
3. Read array elements a[0], a[1], …, a[n-1]
4. For i=0 to n-2 do:
   For j=i+1 to n-1:
      If a[i] > a[j], swap using temp variable.
5. Print the sorted array elements.
6. Stop
```



```c
#include <stdio.h>

int main() {
    int a[100], n, i, j, temp;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    for(i = 0; i < n - 1; i++) {
        for(j = i + 1; j < n; j++) {
            if(a[i] > a[j]) {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }

    printf("Sorted array in ascending order:\n");
    for(i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }

    return 0;
}
```

```
Enter number of elements: 5
Enter 5 elements:
22 74 99 14 46
Sorted array in ascending order:
14 22 46 74 99

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

Example:

Input: Enter 5 elements: 22 74 99 14 46

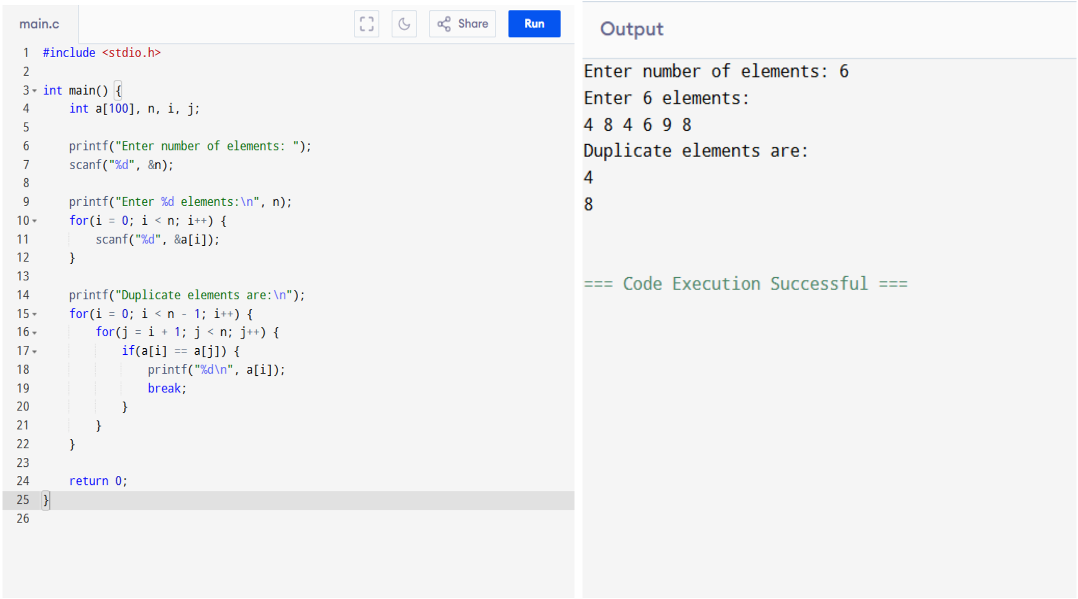
Output: Sorted Array: 14 22 46 74 99

2. Find and Display Duplicate Elements

Aim: To write a program to find and display duplicate elements in a list.

Algorithm Steps:

```
1. Start
2. Read number of elements, n
3. Read array elements
4. For i=0 to n-2:
  For j=i+1 to n-1:
     If a[i] == a[j], print a[i] as duplicate
5. Stop
```



```c
#include <stdio.h>

int main() {
    int a[100], n, i, j;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    printf("Duplicate elements are:\n");
    for(i = 0; i < n - 1; i++) {
        for(j = i + 1; j < n; j++) {
            if(a[i] == a[j]) {
                printf("%d\n", a[i]);
                break;
            }
        }
    }

    return 0;
}
```

```
Enter number of elements: 6
Enter 6 elements:
4 8 4 6 9 8
Duplicate elements are:
4
8


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

Example:

Input: 4 8 4 6 9 8

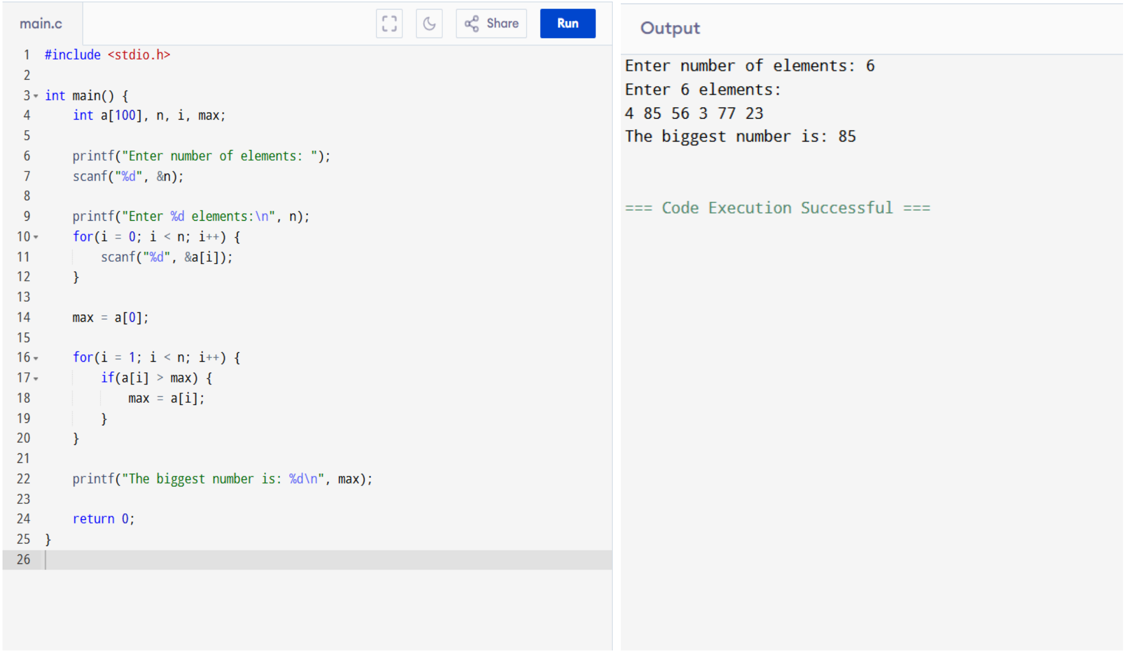
Output: Duplicate elements are: 4 8

3. Biggest Number in a Series

Aim: To write a C program to find the biggest number in a series.

Algorithm Steps:

```
1. Start
2. Read n
3. Read array elements
4. Initialize max = a[0]
5. For i=1 to n-1:
   If a[i] > max, then max = a[i]
6. Print max
7. Stop
```



```c
#include <stdio.h>

int main() {
    int a[100], n, i, max;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    max = a[0];

    for(i = 1; i < n; i++) {
        if(a[i] > max) {
            max = a[i];
        }
    }

    printf("The biggest number is: %d\n", max);

    return 0;
}
```

```
Enter number of elements: 6
Enter 6 elements:
4 85 56 3 77 23
The biggest number is: 85


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

Example:

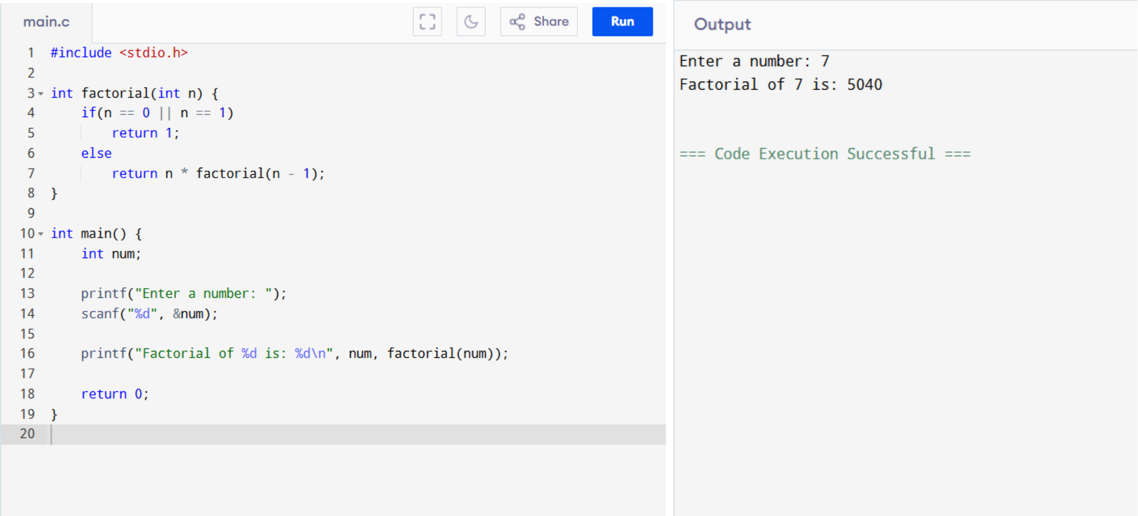
Input: 4 85 56 3 77 23

Output: Biggest number is 85

4. Factorial using Recursion

Aim: To write a C program to find the factorial using recursion.

Algorithm Steps:
1. Start
2. Read n
3. If n==0 or n==1, return 1
4. Else return n * factorial(n-1)
5. Display result
6. Stop



```c
#include <stdio.h>

int factorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}

int main() {
    int num;

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

    printf("Factorial of %d is: %d\n", num, factorial(num));

    return 0;
}
```

```
Enter a number: 7
Factorial of 7 is: 5040


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

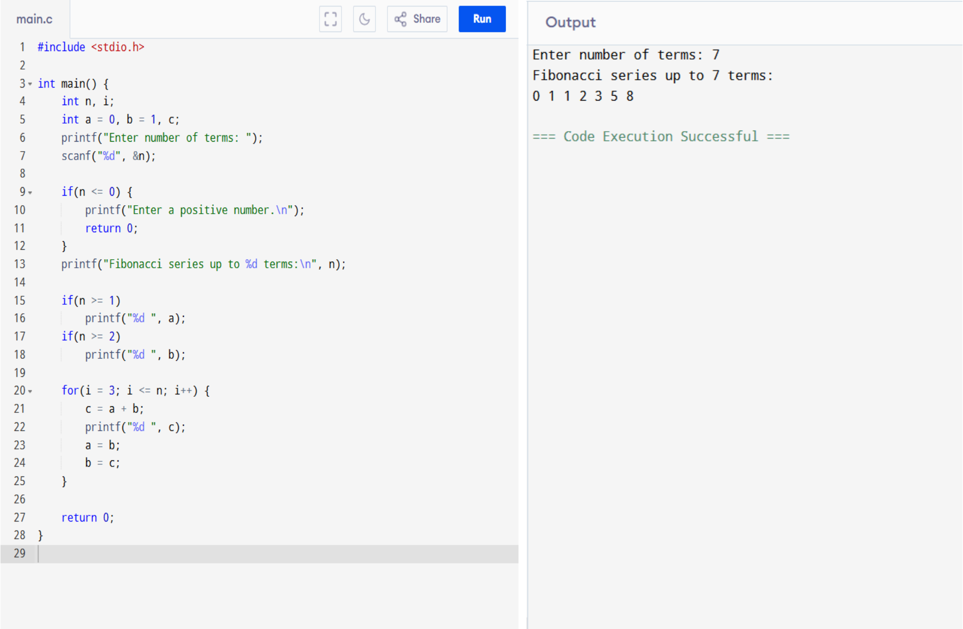
Example:
Input: 7
Output: Factorial = 5040

5. Fibonacci Series

Aim: To write a C program to generate Fibonacci series.

Algorithm Steps:

```
1. Start
2. Read n
3. Initialize a=0, b=1
4. Print a, b
5. For i=3 to n:
   c=a+b; print c; a=b; b=c
6. Stop
```



```c
#include <stdio.h>

int main() {
    int n, i;
    int a = 0, b = 1, c;
    printf("Enter number of terms: ");
    scanf("%d", &n);

    if(n <= 0) {
        printf("Enter a positive number.\n");
        return 0;
    }
    printf("Fibonacci series up to %d terms:\n", n);

    if(n >= 1)
        printf("%d ", a);
    if(n >= 2)
        printf("%d ", b);

    for(i = 3; i <= n; i++) {
        c = a + b;
        printf("%d ", c);
        a = b;
        b = c;
    }

    return 0;
}
```

```
Enter number of terms: 7
Fibonacci series up to 7 terms:
0 1 1 2 3 5 8

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

Example:

Input: 7

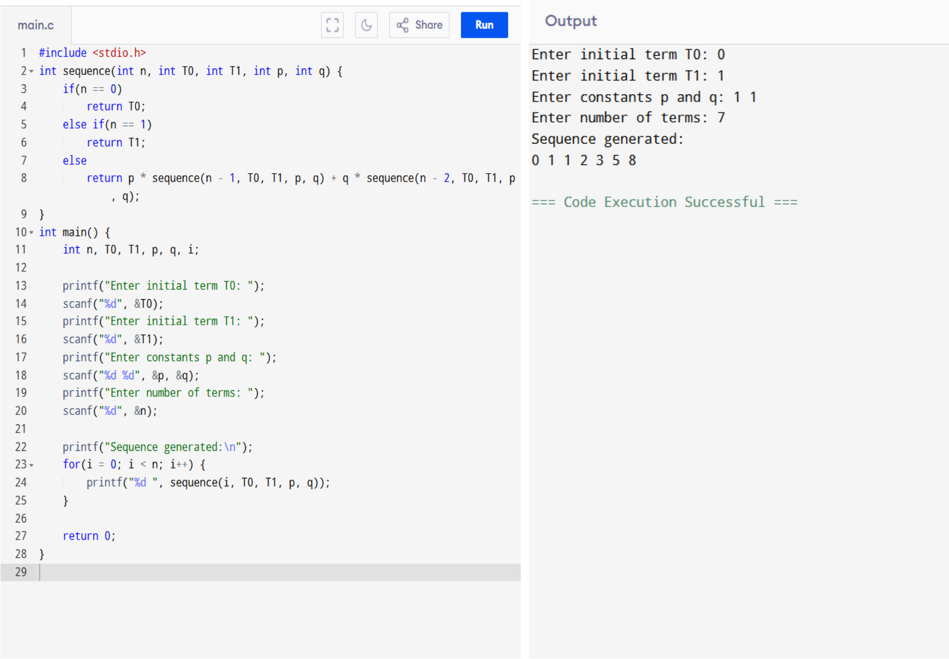
Output: 0 1 1 2 3 5 8

6. Two-order Homogeneous Linear Recursion

Aim: To find second-order homogeneous linear recursion using recursion.

Algorithm Steps:

```
1. Start
2. Read P, Q, T0, T1, n
3. Define recursive function T(n):
   If n=0 return T0
   If n=1 return T1
   Else return P*T(n-1)+Q*T(n-2)
4. Print all terms
5. Stop
```



```c
#include <stdio.h>
int sequence(int n, int T0, int T1, int p, int q) {
    if(n == 0)
        return T0;
    else if(n == 1)
        return T1;
    else
        return p * sequence(n - 1, T0, T1, p, q) + q * sequence(n - 2, T0, T1, p, q);
}
int main() {
    int n, T0, T1, p, q, i;

    printf("Enter initial term T0: ");
    scanf("%d", &T0);
    printf("Enter initial term T1: ");
    scanf("%d", &T1);
    printf("Enter constants p and q: ");
    scanf("%d %d", &p, &q);
    printf("Enter number of terms: ");
    scanf("%d", &n);

    printf("Sequence generated:\n");
    for(i = 0; i < n; i++) {
        printf("%d ", sequence(i, T0, T1, p, q));
    }

    return 0;
}
```

```
Enter initial term T0: 0
Enter initial term T1: 1
Enter constants p and q: 1 1
Enter number of terms: 7
Sequence generated:
0 1 1 2 3 5 8

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

Example:

Input: T0=0, T1=1, P=1, Q=1, n=7

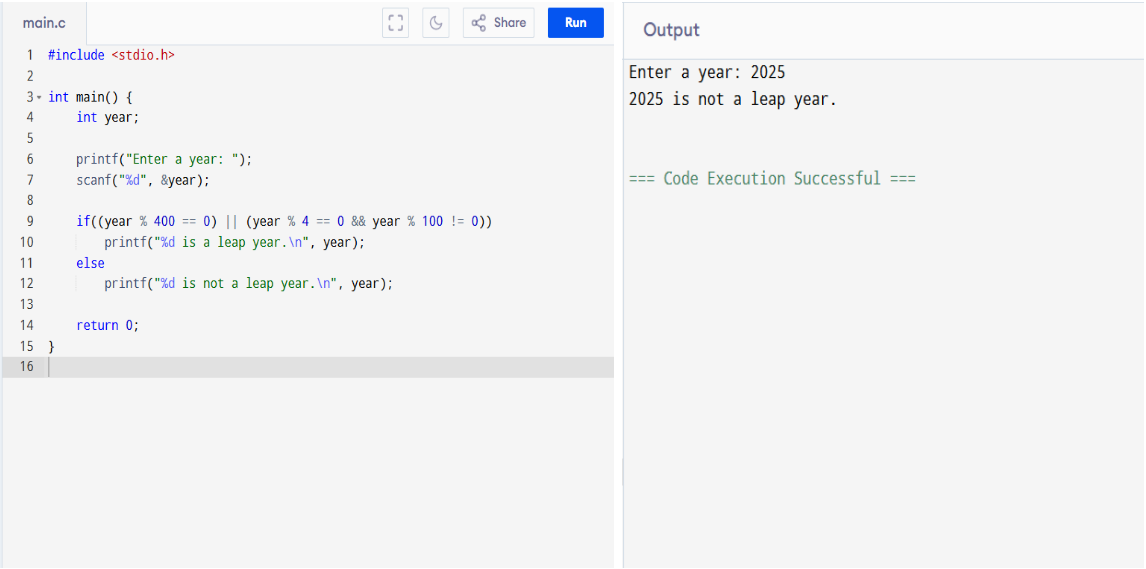
Output: 0 1 1 2 3 5 8

7. Leap Year Check

Aim: To check if a year is a leap or not.

Algorithm Steps:

1. Start
2. Read year
3. If year%400==0 → Leap Year
4. Else if year%100==0 → Not Leap Year
5. Else if year%4==0 → Leap Year
6. Else → Not Leap Year
7. Stop



```c
#include <stdio.h>

int main() {
    int year;

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

    if((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0))
        printf("%d is a leap year.\n", year);
    else
        printf("%d is not a leap year.\n", year);

    return 0;
}
```

```
Enter a year: 2025
2025 is not a leap year.


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

Example:

Input: 2025

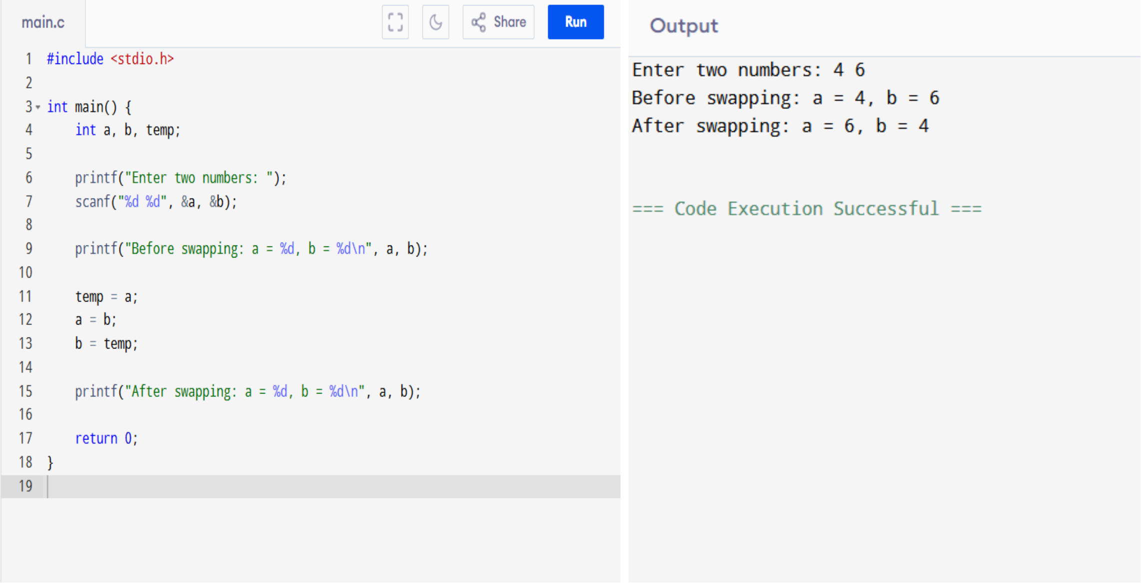
Output: 2025 is not a leap year

8. Swapping Two Numbers

Aim: To swap two numbers.

Algorithm Steps:

1. Start
2. Read a, b
3. temp=a; a=b; b=temp
4. Print swapped values
5. Stop



```c
#include <stdio.h>

int main() {
    int a, b, temp;

    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

    printf("Before swapping: a = %d, b = %d\n", a, b);

    temp = a;
    a = b;
    b = temp;

    printf("After swapping: a = %d, b = %d\n", a, b);

    return 0;
}
```

```
Enter two numbers: 4 6
Before swapping: a = 4, b = 6
After swapping: a = 6, b = 4


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

Example:

Input: 4 6

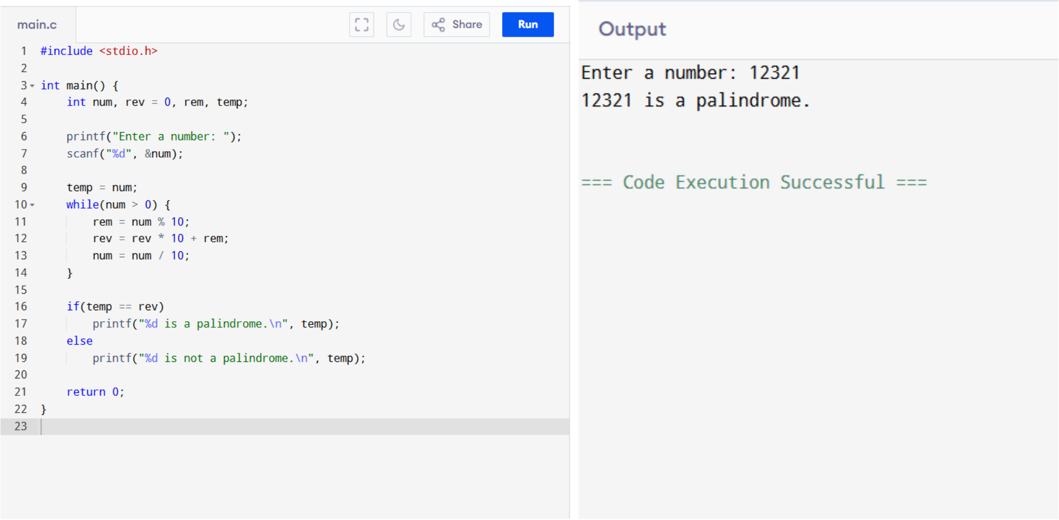
Output: Before swap: a=4 b=6 → After swap: a=6 b=4

9. Palindrome Check

Aim: To check if a number is palindrome.

Algorithm Steps:

```
1. Start
2. Read num
3. rev=0, temp=num
4. While num>0: rem=num%10; rev=rev*10+rem; num=num/10
5. If temp==rev → Palindrome else Not
6. Stop
```



```c
#include <stdio.h>

int main() {
    int num, rev = 0, rem, temp;

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

    temp = num;
    while(num > 0) {
        rem = num % 10;
        rev = rev * 10 + rem;
        num = num / 10;
    }

    if(temp == rev)
        printf("%d is a palindrome.\n", temp);
    else
        printf("%d is not a palindrome.\n", temp);

    return 0;
}
```

```
Enter a number: 12321
12321 is a palindrome.


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

Example:

Input: 12321

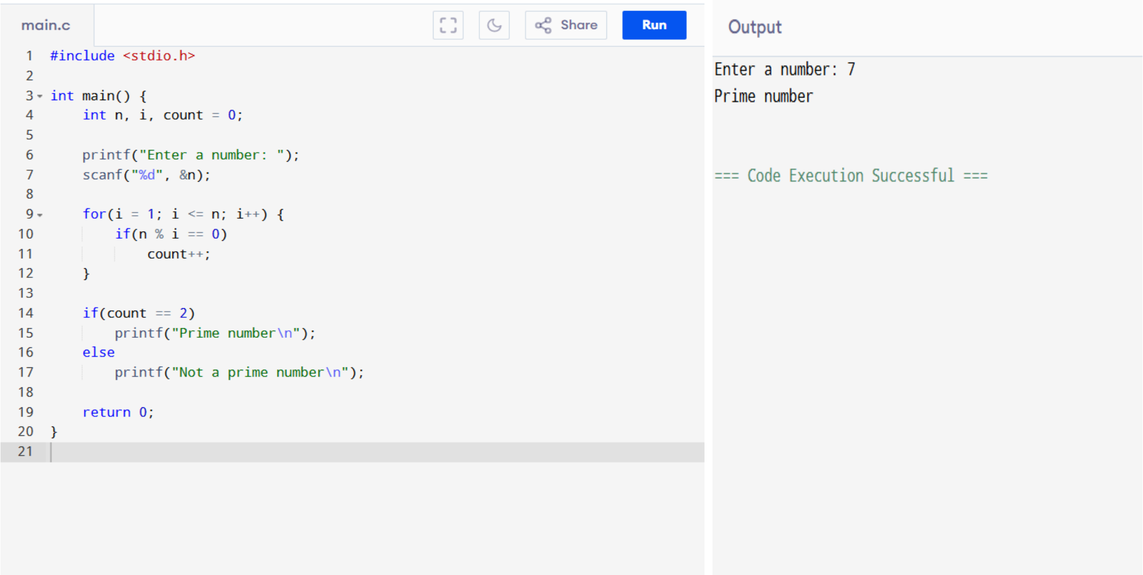
Output: Palindrome

10. Prime Number Check

Aim: To check if a number is prime.

Algorithm Steps:

```
1. Start
2. Read n
3. count=0
4. For i=1 to n:
   If n%i==0 count++
5. If count==2 → Prime else Not Prime
6. Stop
```



```c
#include <stdio.h>

int main() {
    int n, i, count = 0;

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

    for(i = 1; i <= n; i++) {
        if(n % i == 0)
            count++;
    }

    if(count == 2)
        printf("Prime number\n");
    else
        printf("Not a prime number\n");

    return 0;
}
```

```
Enter a number: 7
Prime number


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

Example:

Input: 7

Output: Prime Number