BOOK 1 OF 8

Absolute Beginner

Your First Steps into C++ Programming

120 Problems with Complete C++ Solutions

Statement · Examples · Constraints · Hint · Solution · Complexity · Explanation

The journey begins here. These 120 problems assume nothing: they take you from printing your very first line of output to confidently using variables, conditions, loops, strings, and arrays. Every problem is small, self-contained, and fully solved, so you build real fluency one idea at a time.

TOPICS COVERED

Output · Variables · Input · Math Basics · Conditionals · Loops · Strings · Arrays

Table of Contents

#034	Minimum of Two	Easy
#035	Maximum of Three	Easy
#036	Absolute Value	Easy
#037	Leap Year	Easy
#038	Grade from Score	Easy
#039	Sign of a Number	Easy
#040	Divisible by Both 3 and 5	Easy
#041	Is It a Vowel?	Easy
#042	Three Equal Numbers	Easy
#043	Within a Range	Easy
#044	Older of Two People	Easy
#045	Pass or Fail	Easy
#046	Triangle Validity	Easy
#047	Quadrant of a Point	Medium
#048	Even and Positive	Easy
#049	Ticket Price by Age	Easy
#050	Maximum Without if (ternary)	Easy
#051	Buzz Number	Easy
#052	Same Sign	Easy
#053	Discount Eligibility	Easy
#054	Middle of Three	Medium
#055	Right Angle Check	Medium
#056	FizzBuzz for One Number	Easy
#057	Character Case	Easy
#058	Number of Days in a Month	Medium
#059	Compare Three for Equality Count	Medium
#060	Rock Paper Scissors	Medium
#061	Count from 1 to N	Easy
#062	Sum from 1 to N (Loop)	Easy
#063	Print Even Numbers up to N	Easy
#064	Multiplication Table	Easy
#065	Factorial	Easy
#066	Power (Repeated Multiplication)	Easy
#067	Count Digits	Easy
#068	Sum of Digits	Easy

#069	Reverse a Number	Easy
#070	Is the Number Prime?	Medium
#071	Count Divisors	Medium
#072	Fibonacci up to N Terms	Easy
#073	GCD (Euclid)	Medium
#074	LCM of Two Numbers	Medium
#075	Sum of Even and Odd	Easy
#076	Largest Digit	Easy
#077	Count Multiples	Easy
#078	Is Perfect Number	Medium
#079	Print Square Pattern	Easy
#080	Right Triangle Pattern	Easy
#081	Sum of Squares	Easy
#082	Collatz Steps	Medium
#083	Count Set Bits	Medium
#084	Decimal to Binary	Medium
#085	Harmonic-Like Count	Easy
#086	Repeated Digit Sum (Digital Root)	Medium
#087	Count Trailing Zeros of Factorial	Medium
#088	Fast Power (Exponentiation by Squaring)	Medium
#089	Sum of Proper Divisors	Medium
#090	Number Pyramid	Medium
#091	Length of a String	Easy
#092	Print First and Last Character	Easy
#093	Reverse a String	Easy
#094	Count Vowels	Easy
#095	String Palindrome	Easy
#096	To Uppercase	Easy
#097	Count a Specific Character	Easy
#098	Count Words	Medium
#099	Replace a Character	Easy
#100	Concatenate Two Words	Easy
#101	Sum of an Array	Easy
#102	Maximum in an Array	Easy
#103	Minimum in an Array	Easy

#104	Average of an Array	Easy
#105	Linear Search	Easy
#106	Count Occurrences	Easy
#107	Reverse an Array	Easy
#108	Count Even and Odd in Array	Easy
#109	Second Largest	Medium
#110	Sort Three Numbers	Easy
#111	Sum of Two Largest	Medium
#112	Check if Array is Sorted	Easy
#113	Count Elements Above Average	Medium
#114	Frequency of Maximum	Medium
#115	Dot Product	Medium
#116	Running Maximum	Medium
#117	Range of an Array	Easy
#118	Count Distinct (Small Range)	Medium
#119	Most Frequent (Small Range)	Medium
#120	Move Zeros to the End	Medium

#001 Hello, World!

EASY

#output

Write a program that prints the exact text Hello, World! followed by a newline.

Example 1

Input	(no input)
Output	Hello, World!

HINT

Use cout with the << operator to send text to the screen.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

cout is the standard output stream. The << operator writes the text, and endl prints a newline.

#002 Print Your Name

EASY

#output

Print a greeting on two separate lines: the first line says Hello, and the second line says your name (use Alice).

Example 1

Input	(no input)
Output	Hello Alice

HINT

Use two cout statements, or use endl / \n between the words.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello" << endl;
    cout << "Alice" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Each cout writes one line. endl moves the cursor to the next line so the two words do not run together.

#003 Print an Integer

EASY

#variables

Store the integer 42 in a variable and print it.

Example 1

Input	(no input)
Output	42

HINT

Declare an int variable, assign 42, then print the variable.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int x = 42;
    cout << x << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

An int holds whole numbers. We assign 42 to x and print the variable's value.

#004 Print a Decimal Number

EASY

#variables

Store the value 3.14 in a variable of type double and print it.

Example 1

Input	(no input)
Output	3.14

HINT

Use the double type for numbers with a fractional part.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    double pi = 3.14;
    cout << pi << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

double stores real numbers. By default cout prints enough digits to show 3.14 exactly.

#005 Print a Character

EASY

#variables

Store the character A in a char variable and print it.

Example 1

Input	(no input)
Output	A

HINT

A char literal is written in single quotes, like 'A'.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    char c = 'A';
    cout << c << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

A char holds a single character. Character literals use single quotes to distinguish them from strings.

#006 Add Two Constants

EASY

#math

Compute 7 + 5 and print the result.

Example 1

Input	(no input)
Output	12

HINT

You can print the result of an arithmetic expression directly.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << 7 + 5 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The expression 7 + 5 is evaluated first, then its value 12 is sent to cout.

#007 The Four Operations

EASY

#math

For a = 12 and b = 4, print their sum, difference, product, and quotient, each on its own line.

Example 1

Input	(no input)
Output	16 8 48 3

HINT

The operators are +, -, *, and / for division.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 12, b = 4;
    cout << a + b << endl;
    cout << a - b << endl;
    cout << a * b << endl;
    cout << a / b << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Each operator produces one result. Because a and b are ints and 12 is divisible by 4, integer division gives an exact 3.

#008 Remainder (Modulo)

EASY

#math

Print the remainder when 17 is divided by 5.

Example 1

Input	(no input)
Output	2

HINT

The % operator gives the remainder of integer division.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << 17 % 5 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

17 divided by 5 is 3 with a remainder of 2, so 17 % 5 equals 2.

#009 Read and Echo an Integer

EASY

#input

Read one integer from input and print it back.

Example 1

Input	99
Output	99

HINT

Use cin >> to read a value into a variable.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
```

```
    cout << n << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

cin reads the integer typed by the user into n, and cout prints it unchanged.

#010 Sum of Two Inputs

EASY

#input #math

Read two integers and print their sum.

Example 1

Input	3 5
Output	8

CONSTRAINTS

- `-10^9 <= a, b <= 10^9`

HINT

Read both with one cin statement, then add them. Use long long to be safe.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long a, b;
    cin >> a >> b;
    cout << a + b << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

cin reads the two whitespace-separated numbers. long long avoids overflow when both values are near a billion.

#011 Greet by Name

EASY

#input #strings

Read a single word (a name) and print Hello, followed by the name and an exclamation mark.

Example 1

Input	`Sara`
Output	`Hello, Sara!`

HINT

Read into a string with cin, then build the output with <<.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string name;
    cin >> name;
    cout << "Hello, " << name << "!" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

cin reads characters up to the first space into name. The greeting is assembled by chaining << operators.

#012 Area of a Rectangle

EASY

#math #input

Read the width and height of a rectangle and print its area.

Example 1

Input	`5 3`
Output	`15`

CONSTRAINTS

- `1 <= width, height <= 10^4`

HINT

Area = width * height.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long w, h;
    cin >> w >> h;
    cout << w * h << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The area of a rectangle is the product of its two side lengths.

#013 Perimeter of a Rectangle

EASY

#math #input

Read the width and height of a rectangle and print its perimeter.

Example 1

Input	`5 3`
Output	`16`

HINT

Perimeter = 2 * (width + height).

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long w, h;
    cin >> w >> h;
    cout << 2 * (w + h) << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

A rectangle has two widths and two heights, so the perimeter is twice their sum.

#014 Square of a Number

EASY

#math #input

Read an integer and print its square.

Example 1

Input	`6`
Output	`36`

HINT

Multiply the number by itself.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    cout << n * n << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Squaring a number means multiplying it by itself, so n * n gives the answer.

#015 Average of Three Numbers

EASY

#math #input

Read three integers and print their average as an integer (using integer division).

Example 1

Input	4 5 6
Output	5

HINT

Add the three numbers and divide by 3.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b, c;
    cin >> a >> b >> c;
    cout << (a + b + c) / 3 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The average is the sum divided by the count. Integer division drops any fractional part.

#016 Celsius to Fahrenheit

EASY

#math #input

Read a temperature in Celsius (an integer) and print it in Fahrenheit. Use the formula F = C * 9 / 5 + 32.

Example 1

Input	100
Output	212

Example 2

Input	0
Output	32

HINT

Apply the conversion formula directly. Multiply before dividing to keep integer math exact here.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int c;
    cin >> c;
    cout << c * 9 / 5 + 32 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Multiplying by 9 before dividing by 5 keeps the intermediate value an integer and matches the standard formula.

#017 Seconds to Minutes and Seconds

EASY

#math #input

Read a number of seconds and print the equivalent minutes and remaining seconds, separated by a space.

Example 1

Input	130
Output	2 10

HINT

Minutes = total / 60, remaining seconds = total % 60.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int s;
    cin >> s;
    cout << s / 60 << " " << s % 60 << endl;
    return 0;
}
```

```
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Integer division by 60 gives whole minutes, and the modulo gives the leftover seconds.

#018 Last Digit of a Number

EASY

#math #input

Read a non-negative integer and print its last digit.

Example 1

Input	1234
Output	4

HINT

The last digit is the remainder after dividing by 10.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    cout << n % 10 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Taking a number modulo 10 isolates its units digit, since that digit is what remains after removing all the tens.

#019 Remove the Last Digit

EASY

#math #input

Read an integer and print it with its last digit removed.

Example 1

Input	1234
Output	123

HINT

Integer division by 10 shifts every digit one place to the right.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    cout << n / 10 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Dividing an integer by 10 discards the units digit, effectively dropping the last digit.

#020 Swap Two Variables

EASY

#variables #input

Read two integers a and b, swap their values, then print them in the new order.

Example 1

Input	3 7
Output	7 3

HINT

Use a temporary variable, or the built-in swap function.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    int temp = a;
    a = b;
    b = temp;
    cout << a << " " << b << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The temporary variable holds a's original value so it is not lost when b is copied into a.

#021 Sum from 1 to N (Formula)

EASY

#math #input

Read N and print the sum 1 + 2 + ... + N using the closed-form formula.

Example 1

Input	`100`
Output	`5050`

CONSTRAINTS

- `1 <= N <= 10^6`

HINT

The sum of the first N integers equals N * (N + 1) / 2.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    cout << n * (n + 1) / 2 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Gauss's formula pairs the terms to compute the total in constant time, avoiding any loop.

#022 Convert Hours to Minutes

EASY

#math #input

Read a whole number of hours and print the equivalent number of minutes.

Example 1

Input	`3`
Output	`180`

HINT

One hour is 60 minutes.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long h;
    cin >> h;
    cout << h * 60 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Multiplying the number of hours by 60 converts the duration into minutes.

#023 Total Price

EASY

#math #input

Read the unit price of an item and the quantity bought, then print the total cost.

Example 1

Input	`25 4`
Output	`100`

HINT

Total = price * quantity.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long price, qty;
    cin >> price >> qty;
    cout << price * qty << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Multiplying the unit price by the number of items gives the total amount to pay.

#024 Increment and Decrement

EASY

#variables #input

Read an integer n. Print n + 1 and n - 1, separated by a space.

Example 1

Input	`10`
Output	`11 9`

HINT

Just add and subtract 1 from the value.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    cout << n + 1 << " " << n - 1 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Adding one gives the next integer (the successor) and subtracting one gives the previous integer (the predecessor).

#025 Double and Half

EASY

#math #input

Read an even integer n. Print double its value and half its value, separated by a space.

Example 1

Input	8
Output	16 4

HINT

Multiply by 2 and divide by 2.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    cout << n * 2 << " " << n / 2 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Doubling multiplies by two; halving divides by two. Because the input is even, the division is exact.

#026 Sum of Digits of a Two-Digit Number

EASY

#math #input

Read a two-digit number and print the sum of its two digits.

Example 1

Input	47
Output	11

CONSTRAINTS

- `10 <= n <= 99`

HINT

The tens digit is n / 10 and the units digit is n % 10.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    cout << n / 10 + n % 10 << endl;
    return 0;
}
```

Complexity **O(1) time | O(1) space**

EXPLANATION

Dividing by 10 extracts the tens digit and the modulo extracts the units digit; adding them gives the digit sum.

#027 Reverse a Two-Digit Number

EASY

#math #input

Read a two-digit number and print it with its digits reversed.

Example 1

Input	47
Output	74

CONSTRAINTS

- `10 <= n <= 99`

HINT

Multiply the units digit by 10 and add the tens digit.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    cout << (n % 10) * 10 + n / 10 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The old units digit becomes the new tens digit (multiplied by 10) and the old tens digit moves to the units place.

#028 Cube of a Number

EASY

#math #input

Read an integer and print its cube (the number raised to the third power).

Example 1

Input	3
Output	27

HINT

Multiply the number by itself three times.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    cout << n * n * n << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The cube of n is n multiplied by itself twice more, which is n * n * n.

#029 Next Even Number

EASY

#math #input

Read an even integer and print the next even integer.

Example 1

Input	8
Output	10

HINT

Even numbers are two apart.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    cout << n + 2 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Consecutive even numbers differ by 2, so adding 2 to an even number yields the next even number.

#030 Print a Quotation

EASY

#output #strings

Print the line: She said "Hello" to me. — including the double quotation marks.

Example 1

Input	(no input)
Output	She said "Hello" to me.

HINT

Inside a string, a double quote is written with a backslash: \".

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "She said \"Hello\" to me." << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The backslash before a quote is an escape sequence that tells the compiler the quote is part of the text, not the end of the string.

#031 Odd or Even

EASY

#conditionals #input

Read an integer and print Even if it is even, or Odd if it is odd.

Example 1

Input	`4`
Output	`Even`

Example 2

Input	`7`
Output	`Odd`

HINT

A number is even when its remainder modulo 2 is zero.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    if (n % 2 == 0) cout << "Even" << endl;
    else            cout << "Odd" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

n % 2 is 0 for even numbers and 1 (or -1) for odd numbers, so the test distinguishes the two cases.

#032 Positive, Negative, or Zero

EASY

#conditionals #input

Read an integer and print Positive, Negative, or Zero accordingly.

Example 1

Input	`-5`
Output	`Negative`

Example 2

Input	`0`
Output	`Zero`

HINT

Use an if / else if / else chain to cover all three cases.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    if (n > 0)      cout << "Positive" << endl;
    else if (n < 0) cout << "Negative" << endl;
    else            cout << "Zero" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The chain checks greater-than-zero first, then less-than-zero; if neither holds, the value must be exactly zero.

#033 Maximum of Two

EASY

#conditionals #input

Read two integers and print the larger one.

Example 1

Input	3 8
Output	8

HINT

Compare the two values with an if statement.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    if (a > b) cout << a << endl;
    else       cout << b << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

If a is strictly greater than b we print a; otherwise b is at least as large, so we print b.

#034 Minimum of Two

EASY

#conditionals #input

Read two integers and print the smaller one.

Example 1

Input	3 8
Output	3

HINT

Reverse the comparison used for the maximum.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    if (a < b) cout << a << endl;
    else       cout << b << endl;
    return 0;
}
```

Complexity **O(1) time | O(1) space**

EXPLANATION

We print a when it is strictly smaller; otherwise b is the smaller or equal value.

#035 Maximum of Three

EASY

#conditionals #input

Read three integers and print the largest of them.

Example 1

Input	4 9 6
Output	9

HINT

Track a running maximum, or compare all three with nested conditions.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b, c;
    cin >> a >> b >> c;
    int m = a;
    if (b > m) m = b;
    if (c > m) m = c;
    cout << m << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

We start by assuming a is the maximum, then update m whenever a later value is larger.

#036 Absolute Value

EASY

#conditionals #input

Read an integer and print its absolute value (its distance from zero).

Example 1

Input	-12
Output	12

Example 2

Input	5
Output	5

HINT

If the number is negative, print its negation; otherwise print it unchanged.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    if (n < 0) n = -n;
    cout << n << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Negating a negative number makes it positive, which is exactly its absolute value; non-negative values are left alone.

#037 Leap Year

EASY

#conditionals #input

Read a year and print Leap if it is a leap year, otherwise Common. A year is a leap year if it is divisible by 4 but not by 100, except years divisible by 400 are leap years.

Example 1

Input	2000
Output	Leap

Example 2

Input	1900
Output	Common

HINT

Combine the three divisibility rules with logical AND/OR operators.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int y;
    cin >> y;
    if ((y % 4 == 0 && y % 100 != 0) || y % 400 == 0)
        cout << "Leap" << endl;
    else
        cout << "Common" << endl;
    return 0;
}
```

Complexity **O(1) time | O(1) space**

EXPLANATION

The condition encodes the Gregorian rule: divisible by 4 and not by 100, or divisible by 400.

#038 Grade from Score

EASY

#conditionals #input

Read a score from 0 to 100 and print a letter grade: A for 90+, B for 80-89, C for 70-79, D for 60-69, otherwise F.

Example 1

Input	85
Output	B

Example 2

Input	50
Output	F

HINT

Check the boundaries from highest to lowest in an if / else if chain.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int s;
    cin >> s;
    if (s >= 90)      cout << "A" << endl;
    else if (s >= 80) cout << "B" << endl;
    else if (s >= 70) cout << "C" << endl;
    else if (s >= 60) cout << "D" << endl;
    else              cout << "F" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Testing the highest threshold first means each later branch only needs a single lower-bound check.

#039 Sign of a Number

EASY

#conditionals #input

Read an integer and print 1 if it is positive, -1 if negative, or 0 if zero.

Example 1

Input	42
Output	1

HINT

This is the sign (signum) function.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    if (n > 0)      cout << 1 << endl;
    else if (n < 0) cout << -1 << endl;
    else            cout << 0 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The signum function maps every positive number to 1, every negative number to -1, and zero to 0.

#040 Divisible by Both 3 and 5

EASY

#conditionals #input

Read an integer and print Yes if it is divisible by both 3 and 5, otherwise No.

Example 1

Input	`30`
Output	`Yes`

Example 2

Input	`9`
Output	`No`

HINT

A number divisible by both 3 and 5 is divisible by 15.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    if (n % 15 == 0) cout << "Yes" << endl;
    else             cout << "No" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Since 3 and 5 share no common factor, divisibility by both is the same as divisibility by their product, 15.

#041 Is It a Vowel?

EASY

#conditionals #input

Read a single lowercase letter and print Vowel if it is one of a, e, i, o, u, otherwise Consonant.

Example 1

Input	`e`
Output	`Vowel`

Example 2

Input	`k`
Output	`Consonant`

HINT

Compare the character against each vowel using OR.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    char c;
    cin >> c;
    if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')
        cout << "Vowel" << endl;
    else
        cout << "Consonant" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The condition is true when the character matches any of the five vowels; otherwise it is treated as a consonant.

#042 Three Equal Numbers

EASY

#conditionals #input

Read three integers and print Yes if all three are equal, otherwise No.

Example 1

Input	`5 5 5`
Output	`Yes`

Example 2

Input	`5 5 6`
Output	`No`

HINT

If a equals b and b equals c, then all three are equal.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b, c;
    cin >> a >> b >> c;
    if (a == b && b == c) cout << "Yes" << endl;
    else                  cout << "No" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Equality is transitive, so checking a == b and b == c is enough to conclude that a, b, and c are all the same.

#043 Within a Range

EASY

#conditionals #input

Read an integer n. Print In if n is between 10 and 20 inclusive, otherwise Out.

Example 1

Input	15
Output	In

Example 2

Input	25
Output	Out

HINT

Combine two comparisons with the AND operator.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    if (n >= 10 && n <= 20) cout << "In" << endl;
    else                    cout << "Out" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The value lies in the closed interval only when it is at least 10 and at most 20, which the AND captures.

#044 Older of Two People

EASY

#conditionals #input

Read two ages and print Same if they are equal, otherwise print the larger age.

Example 1

Input	30 25
Output	30

Example 2

Input	40 40
Output	Same

HINT

Handle the equality case first, then compare.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    if (a == b)      cout << "Same" << endl;
    else if (a > b)  cout << a << endl;
    else             cout << b << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Checking equality first separates the tie case; the remaining branches simply pick the larger value.

#045 Pass or Fail

EASY

#conditionals #input

Read a score. Print Pass if it is 50 or above, otherwise Fail.

Example 1

Input	50
Output	Pass

Example 2

Input	49
Output	Fail

HINT

Use a single comparison with the threshold 50.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int s;
    cin >> s;
    if (s >= 50) cout << "Pass" << endl;
    else         cout << "Fail" << endl;
```

```
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The pass mark is inclusive, so the >= operator correctly treats exactly 50 as a pass.

#046 Triangle Validity

EASY

#conditionals #input

Read three positive side lengths and print Yes if they can form a triangle, otherwise No. Three sides form a triangle when each side is shorter than the sum of the other two.

Example 1

Input	`3 4 5`
Output	`Yes`

Example 2

Input	`1 2 10`
Output	`No`

HINT

Check all three triangle-inequality conditions joined by AND.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long a, b, c;
    cin >> a >> b >> c;
    if (a + b > c && a + c > b && b + c > a)
        cout << "Yes" << endl;
    else
        cout << "No" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The triangle inequality must hold for every side, so all three sum comparisons must be true simultaneously.

#047 Quadrant of a Point

MEDIUM

#conditionals #input

Read the x and y coordinates of a point (neither is zero) and print which quadrant it lies in: 1, 2, 3, or 4.

Example 1

Input	`3 4`
Output	`1`

Example 2

Input	`-3 -4`
Output	`3`

HINT

Quadrant 1 is x>0,y>0; quadrant 2 is x<0,y>0; quadrant 3 is x<0,y<0; quadrant 4 is x>0,y<0.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int x, y;
    cin >> x >> y;
    if (x > 0 && y > 0)      cout << 1 << endl;
    else if (x < 0 && y > 0) cout << 2 << endl;
    else if (x < 0 && y < 0) cout << 3 << endl;
    else                     cout << 4 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The sign of each coordinate determines the quadrant; the four combinations of signs map to the four numbered regions.

#048 Even and Positive

EASY

#conditionals #input

Read an integer and print Yes only if it is both even and positive, otherwise No.

Example 1

Input	`8`
Output	`Yes`

Example 2

Input	`-8`
Output	`No`

HINT

Two conditions must both hold, so use AND.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    if (n > 0 && n % 2 == 0) cout << "Yes" << endl;
    else                     cout << "No" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The AND requires the number to satisfy both tests at once: strictly greater than zero and divisible by two.

#049 Ticket Price by Age

EASY

#conditionals #input

Read an age. Print 0 for ages under 5, 10 for ages 5 to 17, and 20 for 18 and above.

Example 1

Input	3
Output	0

Example 2

Input	12
Output	10

Example 3

Input	40
Output	20

HINT

Order the age brackets from youngest to oldest.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int age;
    cin >> age;
    if (age < 5)       cout << 0 << endl;
    else if (age < 18) cout << 10 << endl;
    else               cout << 20 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Each branch handles one age bracket; because earlier branches already excluded younger ages, later checks need only one bound.

#050 Maximum Without if (ternary)

EASY

#conditionals #input

Read two integers and print the larger one, using the ternary operator ? :.

Example 1

Input 7 4
Output 7

HINT

The expression condition ? valueIfTrue : valueIfFalse chooses between two values.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << (a > b ? a : b) << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The ternary operator evaluates the condition and yields a when it is true and b otherwise, all in a single expression.

#051 Buzz Number

EASY

#conditionals #input

A buzz number ends in 7 or is divisible by 7. Read an integer and print Buzz or Normal.

Example 1

Input 27
Output Buzz

Example 2

Input 14
Output Buzz

Example 3

Input	13
Output	Normal

HINT

Combine a last-digit check with a divisibility check using OR.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    if (n % 10 == 7 || n % 7 == 0) cout << "Buzz" << endl;
    else                           cout << "Normal" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Either condition is sufficient, so the OR prints Buzz when the number ends in 7 or is a multiple of 7.

#052 Same Sign

EASY

#conditionals #input

Read two non-zero integers and print Yes if they have the same sign, otherwise No.

Example 1

Input	3 9
Output	Yes

Example 2

Input	-3 9
Output	No

HINT

Two numbers share a sign when their product is positive.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long a, b;
    cin >> a >> b;
    if (a * b > 0) cout << "Yes" << endl;
```

```
    else           cout << "No" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The product of two numbers is positive exactly when both are positive or both are negative, i.e. when they share a sign.

#053 Discount Eligibility

EASY

#conditionals #input

A purchase qualifies for a discount if the total is 100 or more. Read the total and print Discount or Full Price.

Example 1

Input	150
Output	Discount

Example 2

Input	80
Output	Full Price

HINT

Compare against the threshold 100.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long total;
    cin >> total;
    if (total >= 100) cout << "Discount" << endl;
    else              cout << "Full Price" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The discount threshold is inclusive, so a total of exactly 100 still qualifies under the >= comparison.

#054 Middle of Three

MEDIUM

#conditionals #input

Read three distinct integers and print the one that is neither the largest nor the smallest.

Example 1

Input	`5 1 3`
Output	`3`

HINT

The middle value is the total minus the maximum minus the minimum.

C++ SOLUTION

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    int a, b, c;
    cin >> a >> b >> c;
    int mx = max(a, max(b, c));
    int mn = min(a, min(b, c));
    cout << a + b + c - mx - mn << endl;
    return 0;
}
```

Complexity **O(1) time | O(1) space**

EXPLANATION

Subtracting both the largest and smallest values from the total of all three leaves exactly the middle value.

#055 Right Angle Check

MEDIUM

#conditionals #input

Read three side lengths of a triangle and print Right if they satisfy the Pythagorean theorem in some order, otherwise Not Right.

Example 1

Input	`3 4 5`
Output	`Right`

Example 2

Input	`2 3 4`
Output	`Not Right`

HINT

Sort the sides so the longest is last, then test a^2 + b^2 == c^2.

C++ SOLUTION

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    long long s[3];
    cin >> s[0] >> s[1] >> s[2];
    sort(s, s + 3);
    if (s[0]*s[0] + s[1]*s[1] == s[2]*s[2])
        cout << "Right" << endl;
    else
        cout << "Not Right" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Sorting puts the hypotenuse last, so the Pythagorean check only needs to be tested once on the two shorter sides.

#056 FizzBuzz for One Number

EASY

#conditionals #input

Read an integer. Print FizzBuzz if divisible by 15, Fizz if divisible by 3, Buzz if divisible by 5, otherwise the number itself.

Example 1

Input	15
Output	FizzBuzz

Example 2

Input	9
Output	Fizz

Example 3

Input	7
Output	7

HINT

Check divisibility by 15 first to capture the combined case.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
```

```cpp
    if (n % 15 == 0)     cout << "FizzBuzz" << endl;
    else if (n % 3 == 0) cout << "Fizz" << endl;
    else if (n % 5 == 0) cout << "Buzz" << endl;
    else                 cout << n << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Testing the 15 case first means the later branches only fire for numbers divisible by 3 alone or 5 alone.

#057 Character Case

EASY

#conditionals #input

Read a single alphabetic character and print Upper if it is an uppercase letter, otherwise Lower.

Example 1

Input	G
Output	Upper

Example 2

Input	g
Output	Lower

HINT

Uppercase letters fall in the range 'A' to 'Z'.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    char c;
    cin >> c;
    if (c >= 'A' && c <= 'Z') cout << "Upper" << endl;
    else                      cout << "Lower" << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Characters are compared by their codes; the uppercase letters occupy a contiguous range from 'A' to 'Z'.

#058 Number of Days in a Month

MEDIUM

#conditionals #input

Read a month number (1-12) for a non-leap year and print how many days it has.

Example 1

Input	2
Output	28

Example 2

Input	4
Output	30

Example 3

Input	1
Output	31

HINT

February has 28; April, June, September, and November have 30; the rest have 31.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int m;
    cin >> m;
    if (m == 2)
        cout << 28 << endl;
    else if (m == 4 || m == 6 || m == 9 || m == 11)
        cout << 30 << endl;
    else
        cout << 31 << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

February is handled separately, the four 30-day months are grouped with OR, and every other month defaults to 31 days.

#059 Compare Three for Equality Count

MEDIUM

#conditionals #input

Read three integers and print how many pairs among them are equal (0, 1, or 3).

Example 1

Input	5 5 5
Output	3

Example 2

Input 5 5 6
Output 1

Example 3

Input 1 2 3
Output 0

HINT

Count each of the three possible pairs separately.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b, c, count = 0;
    cin >> a >> b >> c;
    if (a == b) count++;
    if (b == c) count++;
    if (a == c) count++;
    cout << count << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

There are three unordered pairs; if all values are equal every pair matches (3), if exactly two match only one pair does (1).

#060 Rock Paper Scissors

MEDIUM

#conditionals #input

Read two moves as characters R, P, or S for two players. Print 1 if player 1 wins, 2 if player 2 wins, or 0 for a tie. Rock beats Scissors, Scissors beats Paper, Paper beats Rock.

Example 1

Input R S
Output 1

Example 2

Input P R
Output 1

Example 3

Input R R
Output 0

HINT

List the three winning combinations for player 1; the symmetric cases let player 2 win.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    char a, b;
    cin >> a >> b;
    if (a == b) {
        cout << 0 << endl;
    } else if ((a == 'R' && b == 'S') ||
               (a == 'S' && b == 'P') ||
               (a == 'P' && b == 'R')) {
        cout << 1 << endl;
    } else {
        cout << 2 << endl;
    }
    return 0;
}
```

Complexity **O(1) time | O(1) space**

EXPLANATION

Equal moves tie. The three listed combinations are exactly the cases where player 1 beats player 2; every other non-tie means player 2 wins.

#061 Count from 1 to N

EASY

#loops #input

Read N and print the integers from 1 to N, each on its own line.

Example 1

Input	3
Output	1 2 3

CONSTRAINTS

- `1 <= N <= 10^5`

HINT

Use a for loop with a counter that runs while it is <= N.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
```

```
    cin >> n;
    for (int i = 1; i <= n; i++)
        cout << i << "\n";
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

The loop counter i starts at 1 and increases by one each pass until it exceeds N, printing every value along the way.

#062 Sum from 1 to N (Loop)

EASY

#loops #input

Read N and print the sum 1 + 2 + ... + N by accumulating in a loop.

Example 1

Input	5
Output	15

CONSTRAINTS

- `1 <= N <= 10^6`

HINT

Keep a running total and add each value of the counter to it.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n, sum = 0;
    cin >> n;
    for (long long i = 1; i <= n; i++)
        sum += i;
    cout << sum << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

The accumulator sum starts at zero and grows by i on every iteration, ending with the total of all values.

#063 Print Even Numbers up to N

EASY

#loops #input

Read N and print all even numbers from 2 up to N, separated by spaces.

Example 1

Input	`10`
Output	`2 4 6 8 10`

HINT

Start the counter at 2 and step by 2 each time.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    bool first = true;
    for (int i = 2; i <= n; i += 2) {
        if (!first) cout << " ";
        cout << i;
        first = false;
    }
    cout << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Stepping the counter by 2 visits only even numbers; the first flag controls the spaces so they appear only between values.

#064 Multiplication Table

EASY

#loops #input

Read a number n and print its multiplication table from n x 1 to n x 10 in the form 'n x i = result', one per line.

Example 1

Input	`5`

Output	`5 x 1 = 5`
	`5 x 2 = 10`
	`5 x 3 = 15`
	`5 x 4 = 20`
	`5 x 5 = 25`
	`5 x 6 = 30`
	`5 x 7 = 35`
	`5 x 8 = 40`
	`5 x 9 = 45`
	`5 x 10 = 50`

HINT

Loop i from 1 to 10 and print the formatted line each time.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= 10; i++)
        cout << n << " x " << i << " = " << n * i << "\n";
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The loop runs a fixed ten times, printing the product of n and the current multiplier in the required format.

#065 Factorial

EASY

#loops #input

Read a non-negative integer n and print n! (the product 1 x 2 x ... x n). By definition 0! = 1.

Example 1

Input	5
Output	120

Example 2

Input	0
Output	1

CONSTRAINTS

- `0 <= n <= 20`

HINT

Start a product at 1 and multiply by each value from 1 to n.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long f = 1;
    for (int i = 2; i <= n; i++)
        f *= i;
    cout << f << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Multiplying 1 by every integer from 2 to n builds the factorial; an empty loop (n = 0 or 1) correctly leaves the result at 1.

#066 Power (Repeated Multiplication)

EASY

#loops #input

Read a base and an exponent (both non-negative) and print base raised to that power.

Example 1

Input	`2 10`
Output	`1024`

Example 2

Input	`5 0`
Output	`1`

CONSTRAINTS

- `0 <= exponent <= 62`

HINT

Multiply the result by the base exactly 'exponent' times.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long base;
    int exp;
    cin >> base >> exp;
    long long result = 1;
    for (int i = 0; i < exp; i++)
        result *= base;
    cout << result << endl;
    return 0;
}
```

Complexity O(exponent) time | O(1) space

EXPLANATION

Each iteration multiplies the running result by the base; after 'exponent' iterations the result is base to that power.

#067 Count Digits

EASY

#loops #input

Read a positive integer and print how many digits it has.

Example 1

Input	1234
Output	4

Example 2

Input	7
Output	1

HINT

Repeatedly divide by 10 and count until the number becomes zero.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    int count = 0;
    while (n > 0) {
        n /= 10;
        count++;
    }
    cout << count << endl;
    return 0;
}
```

Complexity **O(d) time | O(1) space**

EXPLANATION

Dividing by 10 removes one digit per step, so the number of divisions needed to reach zero equals the digit count.

#068 Sum of Digits

EASY

#loops #input

Read a non-negative integer and print the sum of its digits.

Example 1

Input	1234
Output	10

HINT

Peel off the last digit with % 10, add it, then drop it with / 10.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    int sum = 0;
    while (n > 0) {
        sum += n % 10;
        n /= 10;
    }
    cout << sum << endl;
    return 0;
}
```

Complexity **O(d) time | O(1) space**

EXPLANATION

On each step the units digit is added to the total and then removed, so all digits are summed by the time the number reaches zero.

#069 Reverse a Number

EASY

#loops #input

Read a non-negative integer and print it with its digits reversed (leading zeros disappear).

Example 1

Input	1234
Output	4321

Example 2

Input	120
Output	21

HINT

Build the reversed number by shifting it left (x10) and appending each peeled digit.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    long long rev = 0;
    while (n > 0) {
        rev = rev * 10 + n % 10;
        n /= 10;
    }
    cout << rev << endl;
    return 0;
}
```

```
14 }
```

Complexity O(d) time | O(1) space

EXPLANATION

Each digit taken from the back of the original is pushed onto the front of rev by multiplying rev by 10 first.

#070 Is the Number Prime?

MEDIUM

#loops #input

Read an integer n (n >= 2) and print Prime if it has no divisors other than 1 and itself, otherwise Not Prime.

Example 1

Input	17
Output	Prime

Example 2

Input	15
Output	Not Prime

CONSTRAINTS

- `2 <= n <= 10^9`

HINT

Test divisors only up to the square root of n.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    bool prime = true;
    for (long long i = 2; i * i <= n; i++) {
        if (n % i == 0) { prime = false; break; }
    }
    cout << (prime ? "Prime" : "Not Prime") << endl;
    return 0;
}
```

Complexity O(sqrt(n)) time | O(1) space

EXPLANATION

If n has a divisor larger than its square root, it must also have a paired smaller one, so checking up to sqrt(n) is sufficient.

#071 Count Divisors

MEDIUM

#loops #input

Read a positive integer and print how many positive divisors it has.

Example 1

Input	12
Output	6

CONSTRAINTS

- `1 <= n <= 10^9`

HINT

When i divides n, both i and n/i are divisors. Count up to sqrt(n) and avoid double-counting a square root.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    long long count = 0;
    for (long long i = 1; i * i <= n; i++) {
        if (n % i == 0) {
            count++;
            if (i != n / i) count++;
        }
    }
    cout << count << endl;
    return 0;
}
```

Complexity **O(sqrt(n)) time | O(1) space**

EXPLANATION

Divisors come in pairs (i, n/i), so we add two per match but only one when i equals n/i, as happens for perfect squares.

#072 Fibonacci up to N Terms

EASY

#loops #input

Read N and print the first N Fibonacci numbers (starting 0, 1) separated by spaces.

Example 1

Input	7
Output	0 1 1 2 3 5 8

CONSTRAINTS

- `1 <= N <= 90`

HINT

Keep the two most recent terms and add them to get the next.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long a = 0, b = 1;
    for (int i = 0; i < n; i++) {
        if (i) cout << " ";
        cout << a;
        long long next = a + b;
        a = b;
        b = next;
    }
    cout << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Only the last two terms are needed to produce the next, so two rolling variables generate the whole sequence in linear time.

#073 GCD (Euclid)

MEDIUM

#loops #input

Read two positive integers and print their greatest common divisor using the Euclidean algorithm.

Example 1

Input	48 36
Output	12

HINT

Repeatedly replace the larger number by the remainder of dividing it by the smaller.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long a, b;
    cin >> a >> b;
    while (b != 0) {
        long long t = a % b;
        a = b;
        b = t;
    }
    cout << a << endl;
```

```
    return 0;
}
```

Complexity O(log(min(a,b))) time | O(1) space

EXPLANATION

Euclid's algorithm relies on the fact that gcd(a, b) equals gcd(b, a mod b); the loop ends when the remainder reaches zero.

#074 LCM of Two Numbers

MEDIUM

#loops #input

Read two positive integers and print their least common multiple.

Example 1

Input	4 6
Output	12

HINT

LCM = a / gcd(a, b) * b. Compute the gcd first, then divide before multiplying to avoid overflow.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

long long gcd(long long a, long long b) {
    while (b) { long long t = a % b; a = b; b = t; }
    return a;
}

int main() {
    long long a, b;
    cin >> a >> b;
    cout << a / gcd(a, b) * b << endl;
    return 0;
}
```

Complexity O(log(min(a,b))) time | O(1) space

EXPLANATION

Because the product of two numbers equals gcd times lcm, dividing one number by the gcd before multiplying keeps the value small and exact.

#075 Sum of Even and Odd

EASY

#loops #input

Read N and print two numbers: the sum of even values and the sum of odd values from 1 to N, separated by a space.

Example 1

Input	5
Output	6 9

HINT

In one loop, add each value to either the even total or the odd total.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n, even = 0, odd = 0;
    cin >> n;
    for (long long i = 1; i <= n; i++) {
        if (i % 2 == 0) even += i;
        else            odd += i;
    }
    cout << even << " " << odd << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

A single pass routes each value to the correct accumulator based on its parity, so both sums are ready after one loop.

#076 Largest Digit

EASY

#loops #input

Read a non-negative integer and print its largest digit.

Example 1

Input	27495
Output	9

HINT

Examine each digit with % 10 and keep the maximum seen.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    int best = 0;
    if (n == 0) { cout << 0 << endl; return 0; }
    while (n > 0) {
```

```
        int d = n % 10;
        if (d > best) best = d;
        n /= 10;
    }
    cout << best << endl;
    return 0;
}
```

Complexity O(d) time | O(1) space

EXPLANATION

Scanning the digits one at a time and remembering the maximum finds the largest digit by the time the number is exhausted.

#077 Count Multiples

EASY

#loops #input

Read N and k. Print how many integers from 1 to N are divisible by k.

Example 1

Input	20 3
Output	6

HINT

You can loop and test divisibility, or simply compute N / k.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n, k, count = 0;
    cin >> n >> k;
    for (long long i = 1; i <= n; i++)
        if (i % k == 0) count++;
    cout << count << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

The loop tests every value and increments the counter whenever it is a multiple of k, giving the total count.

#078 Is Perfect Number

MEDIUM

#loops #input

A perfect number equals the sum of its proper divisors (divisors excluding itself). Read n and print Perfect or Not Perfect.

Example 1

Input	28
Output	Perfect

Example 2

Input	12
Output	Not Perfect

CONSTRAINTS

- `1 <= n <= 10^6`

HINT

Sum every divisor i in 1..n-1 that divides n, then compare to n.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    long long sum = 0;
    for (long long i = 1; i < n; i++)
        if (n % i == 0) sum += i;
    cout << (sum == n ? "Perfect" : "Not Perfect") << endl;
    return 0;
}
```

Complexity **O(n) time | O(1) space**

EXPLANATION

Adding up all divisors strictly less than n and comparing the total to n directly tests the definition of a perfect number.

#079 Print Square Pattern

EASY

#loops #patterns #input

Read n and print an n x n square made of the character *.

Example 1

Input	3
Output	*** *** ***

CONSTRAINTS

- `1 <= n <= 50`

HINT

Use a loop for the rows and an inner loop for the columns.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            cout << "*";
        cout << "\n";
    }
    return 0;
}
```

Complexity O(n^2) time | O(1) space

EXPLANATION

The outer loop produces each row and the inner loop prints n stars per row, forming a solid square.

#080 Right Triangle Pattern

EASY

#loops #patterns #input

Read n and print a left-aligned right triangle of stars: row i contains i stars.

Example 1

Input	`4`

Output	`*`
	`**`
	`***`
	`****`

CONSTRAINTS

- `1 <= n <= 50`

HINT

Let the inner loop run a number of times equal to the current row number.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < i; j++)
            cout << "*";
        cout << "\n";
    }
    return 0;
}
```

```
}
```

Complexity O(n^2) time | O(1) space

EXPLANATION

Because the inner loop count grows with the row index i, each successive row is one star wider, forming a triangle.

#081 Sum of Squares

EASY

#loops #input

Read N and print the sum of squares 1^2 + 2^2 + ... + N^2.

Example 1

Input	3
Output	14

CONSTRAINTS

- `1 <= N <= 10^5`

HINT

Add i*i to a running total for each i.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n, sum = 0;
    cin >> n;
    for (long long i = 1; i <= n; i++)
        sum += i * i;
    cout << sum << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Each iteration contributes the square of the current value to the accumulator, producing the sum of squares.

#082 Collatz Steps

MEDIUM

#loops #input

Read a positive integer n. Repeatedly apply: if even, halve it; if odd, multiply by 3 and add 1. Print how many steps it takes to reach 1.

Example 1

Input	6
Output	8

Example 2

Input	1
Output	0

CONSTRAINTS

- `1 <= n <= 10^6`

HINT

Count iterations of the while loop until n becomes 1.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    int steps = 0;
    while (n != 1) {
        if (n % 2 == 0) n /= 2;
        else            n = 3 * n + 1;
        steps++;
    }
    cout << steps << endl;
    return 0;
}
```

Complexity O(steps) time | O(1) space

EXPLANATION

The Collatz process is applied until the value collapses to 1, and the counter records how many transformations were needed.

#083 Count Set Bits

MEDIUM

#loops #bits #input

Read a non-negative integer and print how many 1s appear in its binary representation.

Example 1

Input	13
Output	3

HINT

Check the lowest bit with & 1, then shift right; repeat until the number is zero.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    unsigned long long n;
    cin >> n;
    int count = 0;
    while (n > 0) {
        count += n & 1;
        n >>= 1;
    }
    cout << count << endl;
    return 0;
}
```

Complexity O(bits) time | O(1) space

EXPLANATION

Masking with 1 reveals whether the lowest bit is set, and shifting right moves to the next bit until all have been counted.

#084 Decimal to Binary

MEDIUM

#loops #bits #input

Read a positive integer and print its binary representation (no leading zeros).

Example 1

Input	13
Output	1101

Example 2

Input	1
Output	1

HINT

Collect remainders mod 2, which come out least-significant first, then reverse.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

int main() {
    long long n;
    cin >> n;
    string bits = "";
    while (n > 0) {
        bits += char('0' + (n % 2));
        n /= 2;
    }
```

```
    reverse(bits.begin(), bits.end());
    cout << bits << endl;
    return 0;
}
```

Complexity O(log n) time | O(log n) space

EXPLANATION

Repeated division by 2 yields the binary digits from least to most significant, so the collected string is reversed before printing.

#085 Harmonic-Like Count

EASY

#loops #input

Read N and print how many numbers from 1 to N are perfect squares.

Example 1

Input	10
Output	3

Example 2

Input	16
Output	4

HINT

Count i while i*i is at most N.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n, count = 0;
    cin >> n;
    for (long long i = 1; i * i <= n; i++)
        count++;
    cout << count << endl;
    return 0;
}
```

Complexity O(sqrt(n)) time | O(1) space

EXPLANATION

The perfect squares up to N are exactly 1^2, 2^2, ... so the count equals the largest i with i*i <= N.

#086 Repeated Digit Sum (Digital Root)

MEDIUM

#loops #input

Read a non-negative integer. Repeatedly replace it by the sum of its digits until a single digit remains, then print that digit.

Example 1

Input	9875
Output	2

HINT

Use an outer loop that keeps reducing while the number has more than one digit.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    while (n >= 10) {
        long long s = 0;
        while (n > 0) { s += n % 10; n /= 10; }
        n = s;
    }
    cout << n << endl;
    return 0;
}
```

Complexity O(d) time per pass | O(1) space

EXPLANATION

The inner loop sums the digits and the outer loop repeats that reduction until only a single-digit value, the digital root, remains.

#087 Count Trailing Zeros of Factorial

MEDIUM

#loops #input

Read n and print the number of trailing zeros in n! without computing the factorial. (Count factors of 5.)

Example 1

Input	25
Output	6

Example 2

Input	10
Output	2

CONSTRAINTS

- `0 <= n <= 10^9`

HINT

Trailing zeros come from factors of 10 = 2 x 5, and fives are the bottleneck. Sum n/5 + n/25 + n/125 + ...

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    long long zeros = 0;
    for (long long p = 5; p <= n; p *= 5)
        zeros += n / p;
    cout << zeros << endl;
    return 0;
}
```

Complexity **O(log n) time | O(1) space**

EXPLANATION

Each multiple of 5 contributes a factor of five, each multiple of 25 an extra one, and so on; summing these counts gives the trailing zeros.

#088 Fast Power (Exponentiation by Squaring)

MEDIUM

#loops #math #input

Read a base and a non-negative exponent and print base^exponent using exponentiation by squaring.

Example 1

Input	`2 20`
Output	`1048576`

Example 2

Input	`3 0`
Output	`1`

CONSTRAINTS

- `result fits in a 64-bit integer`

HINT

Square the base each step and multiply it into the answer whenever the current exponent bit is 1.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    long long base;
```

```
    long long exp;
    cin >> base >> exp;
    long long result = 1;
    while (exp > 0) {
        if (exp & 1) result *= base;
        base *= base;
        exp >>= 1;
    }
    cout << result << endl;
    return 0;
}
```

Complexity O(log exponent) time | O(1) space

EXPLANATION

Writing the exponent in binary lets us reach the answer in a logarithmic number of multiplications by squaring the base at each bit.

#089 Sum of Proper Divisors

MEDIUM

#loops #input

Read a positive integer and print the sum of its proper divisors (all divisors except the number itself).

Example 1

Input	12
Output	16

CONSTRAINTS

- `1 <= n <= 10^9`

HINT

Use the sqrt trick: for each i dividing n, add both i and n/i, then subtract n at the end.

C++ SOLUTION

```
#include <iostream>
using namespace std;

int main() {
    long long n;
    cin >> n;
    long long sum = 0;
    for (long long i = 1; i * i <= n; i++) {
        if (n % i == 0) {
            sum += i;
            if (i != n / i) sum += n / i;
        }
    }
    sum -= n;
    cout << sum << endl;
    return 0;
}
```

Complexity O(sqrt(n)) time | O(1) space

EXPLANATION

Pairing divisors around the square root sums all of them efficiently; subtracting n removes the number itself to leave only proper divisors.

#090 Number Pyramid

MEDIUM

#loops #patterns #input

Read n and print n rows where row i contains the numbers 1 to i with no separators.

Example 1

Input	4
Output	1 12 123 1234

CONSTRAINTS

- 1 <= n <= 9

HINT

The inner loop prints the column index from 1 up to the current row number.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++)
            cout << j;
        cout << "\n";
    }
    return 0;
}
```

Complexity O(n^2) time | O(1) space

EXPLANATION

Each row prints an increasing run of digits from 1 to the row number, so the rows grow one digit longer each time.

#091 Length of a String

EASY

#strings #input

Read a single word and print the number of characters in it.

Example 1

Input	`hello`
Output	`5`

HINT

The string's .size() (or .length()) member gives its length.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s;
    cin >> s;
    cout << s.size() << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

A std::string stores its length, so size() returns the character count immediately without scanning.

#092 Print First and Last Character

EASY

#strings #input

Read a non-empty word and print its first character and its last character separated by a space.

Example 1

Input	`world`
Output	`w d`

HINT

Index 0 is the first character; index size()-1 is the last.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s;
    cin >> s;
    cout << s[0] << " " << s[s.size() - 1] << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

Strings are indexed from zero, so the last valid index is one less than the length.

#093 Reverse a String

EASY

#strings #input

Read a word and print it reversed.

Example 1

Input	`hello`
Output	`olleh`

HINT

Use the reverse function from <algorithm>, or print characters from the back.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

int main() {
    string s;
    cin >> s;
    reverse(s.begin(), s.end());
    cout << s << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

reverse swaps characters from the two ends moving inward, turning the string back to front in place.

#094 Count Vowels

EASY

#strings #input

Read a lowercase word and print how many vowels (a, e, i, o, u) it contains.

Example 1

Input	`education`
Output	`5`

HINT

Loop over each character and test if it is one of the vowels.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s;
    cin >> s;
    int count = 0;
    for (char c : s) {
        if (c=='a'||c=='e'||c=='i'||c=='o'||c=='u') count++;
    }
    cout << count << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

The range-based for loop visits every character once, incrementing the counter whenever it matches a vowel.

#095 String Palindrome

EASY

#strings #input

Read a word and print Yes if it reads the same forwards and backwards, otherwise No.

Example 1

Input	level
Output	Yes

Example 2

Input	hello
Output	No

HINT

Compare the string to its reverse, or use two indices moving toward the center.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s;
    cin >> s;
    int i = 0, j = s.size() - 1;
    bool palindrome = true;
    while (i < j) {
        if (s[i] != s[j]) { palindrome = false; break; }
        i++; j--;
    }
    cout << (palindrome ? "Yes" : "No") << endl;
```

```
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Two pointers from the ends compare mirrored characters; any mismatch means the word is not a palindrome.

#096 To Uppercase

EASY

#strings #input

Read a lowercase word and print it in uppercase.

Example 1

Input	hello
Output	HELLO

HINT

The toupper function converts a single character; apply it to each one.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

int main() {
    string s;
    cin >> s;
    for (char &c : s) c = toupper(c);
    cout << s << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

toupper maps each lowercase letter to its uppercase form; taking the character by reference lets us modify the string in place.

#097 Count a Specific Character

EASY

#strings #input

Read a word and a character, then print how many times that character appears in the word.

Example 1

Input	banana a
Output	3

HINT

Loop through the string and compare each character to the target.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s;
    char target;
    cin >> s >> target;
    int count = 0;
    for (char c : s)
        if (c == target) count++;
    cout << count << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

A single scan compares every character to the target and tallies the matches.

#098 Count Words

MEDIUM

#strings #input

Read a full line of text and print the number of words (sequences separated by single spaces).

Example 1

Input	`the quick brown fox`
Output	`4`

HINT

Read tokens one at a time with the stream operator, which skips whitespace automatically.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string word;
    int count = 0;
    while (cin >> word) count++;
    cout << count << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

The stream extraction operator reads one whitespace-delimited token per loop, so counting iterations counts the words.

#099 Replace a Character

EASY

#strings #input

Read a word, an old character, and a new character. Print the word with every occurrence of the old character replaced by the new one.

Example 1

Input	`balloon l x`
Output	`baxxoon`

HINT

Walk through the string and swap matching characters.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s;
    char oldc, newc;
    cin >> s >> oldc >> newc;
    for (char &c : s)
        if (c == oldc) c = newc;
    cout << s << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Each character equal to the old value is overwritten with the new value as the loop passes over it.

#100 Concatenate Two Words

EASY

#strings #input

Read two words and print them joined together with a space between.

Example 1

Input	`hello world`
Output	`hello world`

HINT

You can join strings with the + operator.

C++ SOLUTION

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string a, b;
    cin >> a >> b;
    cout << a + " " + b << endl;
    return 0;
}
```

Complexity O(n) time | O(n) space

EXPLANATION

The + operator on strings produces a new string containing the left operand followed by the right one.

#101 Sum of an Array

EASY

#arrays #input

Read an integer n followed by n integers, and print their sum.

Example 1

Input	5 1 2 3 4 5
Output	15

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Read each element in a loop and add it to a total.

C++ SOLUTION

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long sum = 0;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        sum += x;
    }
    cout << sum << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Each value is added to the accumulator as it is read, so the full total is ready after the loop without storing the array.

#102 Maximum in an Array

EASY

#arrays #input

Read n integers and print the largest one.

Example 1

Input
```
5
3 7 2 9 4
```

Output
```
9
```

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Track the maximum seen so far, starting from the first element.

C++ SOLUTION

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long best = LLONG_MIN;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (x > best) best = x;
    }
    cout << best << endl;
    return 0;
}
```

Complexity **O(n) time | O(1) space**

EXPLANATION

Initialising the running maximum to the smallest possible value guarantees the first element will replace it, after which only larger values update it.

#103 Minimum in an Array

EASY

#arrays #input

Read n integers and print the smallest one.

Example 1

Input 5
3 7 2 9 4

Output 2

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Track the minimum seen so far.

C++ SOLUTION

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long best = LLONG_MAX;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (x < best) best = x;
    }
    cout << best << endl;
    return 0;
}
```

Complexity **O(n) time | O(1) space**

EXPLANATION

Starting from the largest possible value lets the first element take over as the minimum, then only smaller values overwrite it.

#104 Average of an Array

EASY

#arrays #input

Read n integers and print their average rounded down to an integer.

Example 1

Input 4
2 4 6 9

Output 5

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Sum the values, then divide the total by n.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long sum = 0;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        sum += x;
    }
    cout << sum / n << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

The integer division of the total by the count yields the floor of the true average.

#105 Linear Search

EASY

#arrays #input

Read n integers, then a target value. Print the index (0-based) of the first occurrence of the target, or -1 if it is absent.

Example 1

Input	5 10 20 30 40 50 30
Output	2

Example 2

Input	3 1 2 3 9
Output	-1

HINT

Scan from the start and stop at the first match.

C++ SOLUTION

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<long long> a(n);
```

```
    for (auto &x : a) cin >> x;
    long long target;
    cin >> target;
    int idx = -1;
    for (int i = 0; i < n; i++) {
        if (a[i] == target) { idx = i; break; }
    }
    cout << idx << endl;
    return 0;
}
```

Complexity O(n) time | O(n) space

EXPLANATION

Linear search inspects elements in order and returns the position of the first equal element, or -1 if the loop finds none.

#106 Count Occurrences

EASY

#arrays #input

Read n integers and a target value, then print how many times the target appears.

Example 1

Input	6 1 2 2 3 2 4 2
Output	3

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Increment a counter every time an element equals the target.

C++ SOLUTION

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long a[100000];
    for (int i = 0; i < n; i++) cin >> a[i];
    long long target;
    cin >> target;
    int count = 0;
    for (int i = 0; i < n; i++)
        if (a[i] == target) count++;
    cout << count << endl;
    return 0;
}
```

Complexity O(n) time | O(n) space

EXPLANATION

A single pass counts every element equal to the target, giving its frequency in the array.

#107 Reverse an Array

EASY

#arrays #input

Read n integers and print them in reverse order, separated by spaces.

Example 1

Input
```
5
1 2 3 4 5
```

Output
```
5 4 3 2 1
```

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Store the values, then print from the last index down to the first.

C++ SOLUTION

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<long long> a(n);
    for (auto &x : a) cin >> x;
    for (int i = n - 1; i >= 0; i--) {
        cout << a[i];
        if (i) cout << " ";
    }
    cout << endl;
    return 0;
}
```

Complexity **O(n) time | O(n) space**

EXPLANATION

Iterating from the last index toward the first outputs the elements in the opposite order to how they were read.

#108 Count Even and Odd in Array

EASY

#arrays #input

Read n integers and print two counts: how many are even and how many are odd, separated by a space.

Example 1

Input 5
1 2 3 4 6

Output 3 2

HINT

Test each element's parity and increment the matching counter.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int even = 0, odd = 0;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (x % 2 == 0) even++;
        else            odd++;
    }
    cout << even << " " << odd << endl;
    return 0;
}
```

Complexity **O(n) time | O(1) space**

EXPLANATION

Each value increments either the even or odd counter according to its remainder modulo 2.

#109 Second Largest

MEDIUM

#arrays #input

Read n integers (n >= 2, values may repeat) and print the second largest distinct value. If there is no second distinct value, print the largest.

Example 1

Input 5
3 7 7 5 1

Output 5

Example 2

Input 3
4 4 4

Output 4

CONSTRAINTS

- 2 <= n <= 10^5

HINT

Track the largest and the largest value strictly below it in one pass.

C++ SOLUTION

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long first = LLONG_MIN, second = LLONG_MIN;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (x > first) { second = first; first = x; }
        else if (x < first && x > second) { second = x; }
    }
    cout << (second == LLONG_MIN ? first : second) << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Maintaining the top two distinct values as we scan avoids sorting; if no smaller distinct value ever appears, we fall back to the maximum.

#110 Sort Three Numbers

EASY

#arrays #input

Read three integers and print them in non-decreasing order, separated by spaces.

Example 1

Input	5 1 3
Output	1 3 5

HINT

Place them in an array and use the sort function.

C++ SOLUTION

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    int a[3];
    cin >> a[0] >> a[1] >> a[2];
    sort(a, a + 3);
    cout << a[0] << " " << a[1] << " " << a[2] << endl;
    return 0;
}
```

Complexity O(1) time | O(1) space

EXPLANATION

The standard sort arranges the three values in ascending order, which are then printed in sequence.

#111 Sum of Two Largest

MEDIUM

#arrays #input

Read n integers (n >= 2) and print the sum of the two largest values (by position, duplicates allowed).

Example 1

Input	5 1 5 3 9 7
Output	16

CONSTRAINTS

- 2 <= n <= 10^5

HINT

Keep the two highest values seen so far as you read.

C++ SOLUTION

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long m1 = LLONG_MIN, m2 = LLONG_MIN;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (x > m1) { m2 = m1; m1 = x; }
        else if (x > m2) { m2 = x; }
    }
    cout << m1 + m2 << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Tracking the top two values during the scan gives the two largest without sorting; their sum is then printed.

#112 Check if Array is Sorted

EASY

#arrays #input

Read n integers and print Yes if they are in non-decreasing order, otherwise No.

Example 1

Input
```
4
1 2 2 5
```
Output
```
Yes
```

Example 2

Input
```
4
1 3 2 5
```
Output
```
No
```

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Compare each element with the previous one as you read.

C++ SOLUTION

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long prev = LLONG_MIN;
    bool sorted = true;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (x < prev) sorted = false;
        prev = x;
    }
    cout << (sorted ? "Yes" : "No") << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

If any element is smaller than the one before it, the order is broken; otherwise the sequence is sorted.

#113 Count Elements Above Average

MEDIUM

#arrays #input

Read n integers and print how many of them are strictly greater than the average (use a real-valued average).

Example 1

Input 5
1 2 3 4 10
Output 1

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

You need two passes: one to find the sum/average, one to count.

C++ SOLUTION

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<long long> a(n);
    long long sum = 0;
    for (auto &x : a) { cin >> x; sum += x; }
    double avg = (double)sum / n;
    int count = 0;
    for (long long x : a)
        if (x > avg) count++;
    cout << count << endl;
    return 0;
}
```

Complexity O(n) time | O(n) space

EXPLANATION

The first pass computes the exact average as a real number; the second pass counts values that exceed it.

#114 Frequency of Maximum

MEDIUM

#arrays #input

Read n integers and print how many times the maximum value occurs.

Example 1

Input 6
4 9 9 2 9 1
Output 3

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

First find the maximum, then count elements equal to it (two passes).

C++ SOLUTION

```cpp
#include <iostream>
#include <vector>
#include <climits>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<long long> a(n);
    long long best = LLONG_MIN;
    for (auto &x : a) { cin >> x; if (x > best) best = x; }
    int count = 0;
    for (long long x : a)
        if (x == best) count++;
    cout << count << endl;
    return 0;
}
```

Complexity O(n) time | O(n) space

EXPLANATION

After determining the largest value, a second scan tallies how many elements match it.

#115 Dot Product

MEDIUM

#arrays #input

Read n, then two arrays of n integers each. Print their dot product (the sum of the products of corresponding elements).

Example 1

Input	3 1 2 3 4 5 6
Output	32

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Read both arrays, then multiply matching positions and accumulate.

C++ SOLUTION

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<long long> a(n), b(n);
    for (auto &x : a) cin >> x;
    for (auto &x : b) cin >> x;
    long long dot = 0;
```

```
    for (int i = 0; i < n; i++)
        dot += a[i] * b[i];
    cout << dot << endl;
    return 0;
}
```

Complexity O(n) time | O(n) space

EXPLANATION

The dot product pairs the i-th element of each array, multiplies them, and sums all such products.

#116 Running Maximum

MEDIUM

#arrays #input

Read n integers and print the running maximum after each element: position i shows the largest of the first i+1 values.

Example 1

Input	5 3 1 4 1 5
Output	3 3 4 4 5

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Maintain the best value seen so far and print it after reading each element.

C++ SOLUTION

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long best = LLONG_MIN;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (x > best) best = x;
        cout << best;
        if (i < n - 1) cout << " ";
    }
    cout << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Updating and printing the maximum after each input produces the prefix maximum at every position in a single pass.

#117 Range of an Array

EASY

#arrays #input

Read n integers and print the difference between the maximum and minimum values.

Example 1

Input
```
5
3 7 2 9 4
```

Output
```
7
```

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Track both the maximum and minimum in one pass, then subtract.

C++ SOLUTION

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main() {
    int n;
    cin >> n;
    long long mx = LLONG_MIN, mn = LLONG_MAX;
    for (int i = 0; i < n; i++) {
        long long x;
        cin >> x;
        if (x > mx) mx = x;
        if (x < mn) mn = x;
    }
    cout << mx - mn << endl;
    return 0;
}
```

Complexity O(n) time | O(1) space

EXPLANATION

Tracking the largest and smallest values simultaneously lets us report their difference, the spread of the data, after one scan.

#118 Count Distinct (Small Range)

MEDIUM

#arrays #input

Read n integers, each between 0 and 100. Print how many distinct values appear.

Example 1

Input 6
1 2 2 3 1 4

Output 4

CONSTRAINTS

- `1 <= n <= 10^5`
- `0 <= value <= 100`

HINT

Use a boolean seen[] array indexed by the value.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    bool seen[101] = {false};
    int distinct = 0;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        if (!seen[x]) { seen[x] = true; distinct++; }
    }
    cout << distinct << endl;
    return 0;
}
```

Complexity **O(n) time | O(1) space**

EXPLANATION

Because values are bounded, a fixed boolean array can mark each value the first time it is seen, counting distinct entries directly.

#119 Most Frequent (Small Range)

MEDIUM

#arrays #input

Read n integers, each between 0 and 9. Print the value that occurs most often; if several tie, print the smallest such value.

Example 1

Input 7
3 1 3 2 1 3 2

Output 3

CONSTRAINTS

- `1 <= n <= 10^5`
- `0 <= value <= 9`

HINT

Tally counts in an array of size 10, then scan for the maximum count.

C++ SOLUTION

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int freq[10] = {0};
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        freq[x]++;
    }
    int best = 0;
    for (int d = 1; d < 10; d++)
        if (freq[d] > freq[best]) best = d;
    cout << best << endl;
    return 0;
}
```

Complexity **O(n) time | O(1) space**

EXPLANATION

Counting each value in a fixed-size table and scanning from the smallest index for the highest count naturally breaks ties toward the smallest value.

#120 Move Zeros to the End

MEDIUM

#arrays #input

Read n integers and print them with all zeros moved to the end, keeping the relative order of the non-zero values.

Example 1

Input	6 0 1 0 3 12 0
Output	1 3 12 0 0 0

CONSTRAINTS

- `1 <= n <= 10^5`

HINT

Write the non-zero values to the front in order, then fill the rest with zeros.

C++ SOLUTION

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
```

```
    cin >> n;
    vector<long long> a(n);
    for (auto &x : a) cin >> x;
    int pos = 0;
    for (int i = 0; i < n; i++)
        if (a[i] != 0) a[pos++] = a[i];
    while (pos < n) a[pos++] = 0;
    for (int i = 0; i < n; i++) {
        cout << a[i];
        if (i < n - 1) cout << " ";
    }
    cout << endl;
    return 0;
}
```

Complexity O(n) time | O(n) space

EXPLANATION

A write pointer compacts the non-zero values to the front in their original order, after which the remaining positions are filled with zeros.