Q-1: Help fitness manager to Calculate the Body Mass Index (BMI) of clients.

Sample test case:

|  |
| --- |
| Input: Enter weight(kg) and height(m^2) 72.57 1.78  Output: The Body Mass Index (BMI) is: 22.9043 |

Constraints:

* The values of weight(kg) and height(m^2) should be positive integers.

Solution:

#include <iostream>

using namespace std;

int main() {

double weight;

double height;

cout<<"Enter weight(kg) and height(m^2) ";

cin>>weight>>height;

double bmi = weight / (height \* height);

cout << "The Body Mass Index (BMI) is: " << bmi << std::endl;

return 0;

}

Q-2: Write a program to find maximum and minimum of two numbers without using loop and any condition.

Sample test case:

|  |
| --- |
| Input: 17 19  Output: max = 19  min = 17 |

Solution:

#include<iostream>

#include <cstdlib> // Header file for abs() method

using namespace std;

int main()

{

int a, b;

cin >> a >> b; // Read two integer values 'a' and 'b' from the user.

// Calculate the maximum and minimum of the two input integers 'a' and 'b'.

// The formula for calculating the maximum of two numbers without using the max() function is:

// max = ((a + b) + abs(a - b)) / 2

int maxi = ((a + b) + abs(a - b)) / 2; // Calculate the maximum of 'a' and 'b'.

int mini = ((a + b) - abs(a - b)) / 2; // Calculate the minimum of 'a' and 'b'.

cout << "max = " << maxi << endl; // Print the calculated maximum value.

cout << "min = " << mini; // Print the calculated minimum value.

return 0;

}

Q-3: Create program to check if two numbers are equal without using arithmetic operators comparison operators.

Sample test case:

|  |
| --- |
| Input: 5 5  Output: x is equal to y |
|  |

Solution:

#include <iostream>

using namespace std;

int main()

{

int x;

int y;

cin >> x >> y; // Read two integer values (x and y) from the user.

// The expression "x ^ y" performs the bitwise XOR operation between x and y.

// If the result of the XOR operation is zero, it means both x and y have the same binary representation,

// and thus, they are equal.

// If the result of the XOR operation is non-zero, it means x and y have different binary representations,

// and they are not equal.

if (!(x ^ y))

cout << " x is equal to y "; // If x is equal to y, print this message.

else

cout << " x is not equal to y "; // If x is not equal to y, print this message.

return 0;

}

Q-4: You will be given an integer n, your task is to return the sum of all natural number less than or equal to n.

As the answer could be very large, return answer modulo 109+7.

Sample test case:

|  |
| --- |
| Input: 5  Output: Sum of natural numbers up to 5 (mod 10^9 + 7): 15 |

Constraints:

* The value of n should be positive integers.

Solution:

#include <iostream>

const int MOD = 1000000007;

int sumOfNaturalNumbers(int n) {

long long sum = 0;

// Calculate the sum using the formula (n \* (n + 1) / 2)

sum = (static\_cast<long long>(n) \* (n + 1)) / 2;

// Apply modulo operation to the sum

sum %= MOD;

return static\_cast<int>(sum);

}

int main() {

int n = 500;

int result = sumOfNaturalNumbers(n);

std::cout << "Sum of natural numbers up to " << n << " (mod 10^9 + 7): " << result << std::endl;

return 0;

}

Q-5: Given three integers 'A' denoting the first term of an arithmetic sequence , 'C' denoting the common difference

of an arithmetic sequence and an integer 'B'. you need to tell whether 'B' exists in the arithmetic sequence or not.

Return 1 if B is present in the sequence. Otherwise, returns 0.

Sample test case:

|  |
| --- |
| Input: A = 1, B = 2, C = 3  Output: 2 is not present in the arithmetic sequence. |

Solution:

#include <iostream>

int isPresentInArithmeticSequence(int A, int C, int B) {

// Check if B is equal to the first term A

if (B == A)

return 1;

// Check if B is reachable from A by adding multiples of the common difference C

if ((B - A) % C == 0 && (B - A) / C > 0)

return 1;

return 0;

}

int main() {

int A = 1; // First term of the arithmetic sequence

int C = 2; // Common difference of the arithmetic sequence

int B = 3; // Integer to check

int result = isPresentInArithmeticSequence(A, C, B);

if (result == 1)

std::cout << B << " is present in the arithmetic sequence." << std::endl;

else

std::cout << B << " is not present in the arithmetic sequence." << std::endl;

return 0;

}

Q-6: As a part of conversion calculator you are tasked to Convert temperature from Fahrenheit to Celsius

Sample test case:

|  |
| --- |
| Input: 75.0  Output: The temperature in Celsius is: 23.8889 |

Output:

#include <iostream>

int main() {

double fahrenheit = 75.0;

double celsius = (fahrenheit - 32) \* 5 / 9;

std::cout << "The temperature in Celsius is: " << celsius << std::endl;

return 0;

}

Q-7: Write a program to swap two numbers using bitwise XOR operator.

Sample test case:

|  |
| --- |
| Before swap: a = 10, b = 20  After swap: a = 20, b = 10 |

Constraints:

* Don’t use third variable for swapping.

Output:

#include <iostream>

using namespace std;

int main() {

int a = 10; // Initialize variable 'a' with value 10

int b = 20; // Initialize variable 'b' with value 20

cout << "Before swap: a = " << a << ", b = " << b << endl; // Print the values of 'a' and 'b' before swapping

a = a ^ b; // Perform bitwise XOR operation between 'a' and 'b' and assign the result to 'a'

b = a ^ b; // Perform bitwise XOR operation between 'a' and 'b' (now containing the previous value of 'a') and assign the result to 'b'

a = a ^ b; // Perform bitwise XOR operation between 'a' (now containing the previous value of 'b') and 'b' (now containing the previous value of 'a') and assign the result to 'a'

cout << "After swap: a = " << a << ", b = " << b << endl; // Print the values of 'a' and 'b' after swapping

return 0;

}

Q-8: In a mathematics class, the teacher challenges the students to find all Armstrong numbers between 1 and 1000.

An Armstrong number (also known as a narcissistic number) is a number that is equal to the sum of its own digits each raised to the power of the number of digits in the number.

For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153.

Write a program to help students.

Sample test case:

|  |
| --- |
| All the Armstrong numbers between 1 to 1000 : 1 2 3 4 5 6 7 8 9 153 370 371 407 |

Solution:

#include <iostream>

using namespace std;

int main()

{

int ord1, ord2, ord3, total\_sum;

cout << "All the Armstrong numbers between 1 to 1000 : ";

// Loop which will run from 1 to 1000

for (int num = 1; num <= 1000; ++num)

{

// All the single-digit numbers are

// armstrong number.

if (num <= 9)

{

cout << num << " ";

}

else

{

ord1 = num % 10;

ord2 = (num % 100 - ord1) / 10;

ord3 = (num % 1000 - ord2) / 100;

total\_sum = ((ord1 \* ord1 \* ord1) +

(ord2 \* ord2 \* ord2) +

(ord3 \* ord3 \* ord3));

if (total\_sum == num)

{

cout << num << " ";

}

}

}

return 0;

}

Q-9: Generate a magical number which is equal to the sum of digits of a given number.

Sample test case:

|  |
| --- |
| Input: 549  Output: Sum of digits: 18 |

Constraints:

* Consider positive numbers only.

Solution:

#include <iostream>

// Function to calculate the sum of digits in a number

int sumOfDigits(int n) {

int sum = 0;

// Iterate until the number becomes 0

while (n != 0) {

// Get the rightmost digit of the number

int digit = n % 10;

// Add the digit to the sum

sum += digit;

// Remove the rightmost digit from the number

n /= 10;

}

// Return the sum of digits

return sum;

}

int main() {

int n;

// Prompt the user to enter a number

std::cout << "Enter a number: ";

std::cin >> n;

// Calculate the sum of digits

int sum = sumOfDigits(n);

// Print the sum of digits

std::cout << "Sum of digits: " << sum << std::endl;

return 0;

}

Q-10: Alice and Bob are participating in a coding competition. The challenge they are facing is to check whether a given string is a palindrome or not.

Write a program to help them.

Sample test case:

|  |
| --- |
| Input: mom  Output: mom is a palindrome. |

Solution:

#include <iostream>

#include <string>

using namespace std;

// Function to check if a string is a palindrome

bool isPalindrome(const std::string& str) {

int start = 0;

int end = str.length() - 1;

// Iterate over the string from both ends

while (start < end) {

// If characters at the start and end positions don't match, it's not a palindrome

if (str[start] != str[end])

return false;

// Move to the next characters

start++;

end--;

}

// If the loop completes without returning false, the string is a palindrome

return true;

}

int main() {

string str;

// Prompt the user to enter a string

cout << "Enter a string: ";

cin >> str;

// Check if the string is a palindrome

if (isPalindrome(str))

cout << str << " is a palindrome." << std::endl;

else

cout << str << " is not a palindrome." << std::endl;

return 0;

}

Q-11: The teacher has a list of exam scores (an array of integers) from a recent test.

The scores need to be sorted in ascending order, from the lowest score to the highest score.

Implement the bubble sort algorithm to sort an array of integers in ascending order.

Sample test case:

|  |
| --- |
| Input: 3 8 98 76 32 31  Output: Sorted array: 3 8 31 32 76 98 |

Solution:

#include <iostream>

// Function to sort an array using bubble sort algorithm

void bubbleSort(int arr[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

// Compare adjacent elements and swap them if they are in the wrong order

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main() {

int size;

// Prompt the user to enter the size of the array

std::cout << "Enter the size of the array: ";

std::cin >> size;

int arr[size];

// Prompt the user to enter the elements of the array

std::cout << "Enter the elements of the array: ";

for (int i = 0; i < size; i++) {

std::cin >> arr[i];

}

// Sort the array using bubble sort

bubbleSort(arr, size);

// Print the sorted array

std::cout << "Sorted array: ";

for (int i = 0; i < size; i++) {

std::cout << arr[i] << " ";

}

std::cout << std::endl;

return 0;

}

Q-12: Alex and Bella are playing a game with binary numbers. Alex gives Bella a positive integer, and Bella's task is to find the length of the longest sequence of consecutive set bits (1s) in the binary representation of the given integer. Write program to help Bella.

Sample test case:

|  |
| --- |
| Input: 19  Output: Length of longest sequence of consecutive set bits: 2 |

Solution:

#include <iostream>

using namespace std;

int countConsecutiveSetBits(unsigned int num) {

int count = 0; // Variable to store the current count of consecutive set bits

int maxCount = 0; // Variable to store the maximum count of consecutive set bits found

while (num) {

if (num & 1) // If the least significant bit is set (1)

count++; // Increment the count of consecutive set bits

else

count = 0; // Reset the count if the least significant bit is not set (0)

if (count > maxCount) // If the current count is greater than the maximum count

maxCount = count; // Update the maximum count

num >>= 1; // Right shift the number to check the next bit

}

return maxCount; // Return the length of the longest sequence of consecutive set bits

}

int main() {

unsigned int num = 19; // The given integer to count consecutive set bits

int result = countConsecutiveSetBits(num); // Call the function to count the consecutive set bits

cout << "Length of longest sequence of consecutive set bits: " << result << std::endl; // Print the result

return 0;

}

Q-13: In a video game called "Magical Potions," players can collect magical potions to boost their power. Each magical potion has a unique numeric value associated with it.

Players are curious to know if a particular potion can be represented as a power of 2.

You need to help them by creating a program that checks if a given potion's value is a power of 2 or not.

Sample test case:

|  |
| --- |
| Input: 16  Output:16 is a power of 2. |

Solution:

#include <iostream>

using namespace std;

bool isPowerOfTwo(int num) {

if (num <= 0)

return false;

// A number that is a power of two has only one bit set (i.e., it is in the form of 100...00 in binary).

// So, if we perform a bitwise AND operation between the number and its predecessor (num & (num - 1)),

// it should result in zero for power of two numbers.

return (num & (num - 1)) == 0;

}

int main() {

int num = 5;

bool result = isPowerOfTwo(num);

cout << num << " is" << (result ? "" : " not") << " a power of 2." << std::endl;

return 0;

}

Q-14: Alice is learning how to program and wants to practice by writing a program to generate the Fibonacci series. Help Alice to generate Fibonacci series.

Sample test case:

|  |
| --- |
| Input: Enter the number of terms: 8  Output: Fibonacci series: 0 1 1 2 3 5 8 13 |

Constraints:

* Number to terms to generate Fibonacci series up to should be positive.

Solution:

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter the number of terms: ";

cin >> n;

int first = 0, second = 1; //first and second terms of fibonacci series are constant

cout << "Fibonacci series: ";

cout << first << " " << second << " ";

for (int i = 3; i <= n; i++) {

int next = first + second; // add last two terms to generate next term

cout << next << " ";

first = second;

second = next;

}

return 0;

}

Q-15: Given a positive integer N, print count of set bits in it.

Sample test case:

|  |
| --- |
| Input: N = 6  Output: 2 |

Constraints:

* Integer N should be positive only.

Solution:

#include <iostream>

int countSetBits(int num) {

int count = 0;

while (num > 0) {

// Use bitwise AND with 1 to check the rightmost bit

if (num & 1) {

count++;

}

// Right shift the number by 1 to check the next bit

num >>= 1;

}

return count;

}

int main() {

int num = 25;

int result = countSetBits(num);

std::cout << "Number of set bits in " << num << " is: " << result << std::endl;

return 0;

}