Q-1: 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-2: Program to find the maximum difference between two elements in an array,

where the larger element appears after the smaller element:

Sample test case:

|  |
| --- |
| Input: {2, 3, 10, 6, 4, 8, 1}  Output: Maximum Difference: 8 |

Solution:

#include <iostream>

// Function to find the maximum difference between two elements in an array

int findMaxDifference(int arr[], int size) {

int minElement = arr[0];

int maxDiff = arr[1] - arr[0];

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

if (arr[i] - minElement > maxDiff) {

maxDiff = arr[i] - minElement;

}

if (arr[i] < minElement) {

minElement = arr[i];

}

}

return maxDiff;

}

int main() {

int arr[] = {2, 3, 10, 6, 4, 8, 1};

int size = sizeof(arr) / sizeof(arr[0]);

int maxDifference = findMaxDifference(arr, size);

std::cout << "Maximum Difference: " << maxDifference << std::endl;

return 0;

}

Q-3: For an integer N find the number of trailing zeroes in N!.Sample test case:

|  |
| --- |
| Input: 4  Output: The number of trailing zeroes in 4! is: 0 |
| Solution:  #include <iostream>  int countTrailingZeroes(int N) {  int count = 0;  // Divide N by powers of 5 and count the number of multiples of 5  for (int i = 5; N / i >= 1; i \*= 5) {  count += N / i;  }  return count;  }  int main() {  int N = 10;  int zeroes = countTrailingZeroes(N);  std::cout << "The number of trailing zeroes in " << N << "! is: " << zeroes << std::endl;  return 0;  } |

Q-4: Students are arranged in matrix form for assembly. Help coach to find the maximum height student in each row of a students.

Sample test case:

|  |
| --- |
| Input: {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}  Output:  Maximum element in row 1: 3  Maximum element in row 2: 6  Maximum element in row 3: 9 |

Solution:

#include <iostream>

// Function to find the maximum element in each row of a matrix using pointer arithmetic

void findMaxInRows(int\* matrix, int rows, int cols) {

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

int maxElement = \*matrix;

for (int j = 1; j < cols; j++) {

if (\*(matrix + i \* cols + j) > maxElement) {

maxElement = \*(matrix + i \* cols + j);

}

}

std::cout << "Maximum element in row " << i + 1 << ": " << maxElement << std::endl;

}

}

int main() {

int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int rows = sizeof(matrix) / sizeof(matrix[0]);

int cols = sizeof(matrix[0]) / sizeof(matrix[0][0]);

findMaxInRows(reinterpret\_cast<int\*>(matrix), rows, cols);

return 0;

}