Write a C++ that does the following:
Provided a number of distinct colors needed (2 or more), create a palette of colors that avoid or minimize problematic color combinations

Creating a color palette that avoids or minimizes problematic color combinations involves considering factors such as color contrast, readability, and accessibility. One way to achieve this is by selecting colors that are perceptually distinct. Here's a simple example in C++ using the RGB color space:

cpp Copy code

```cpp
#include <iostream> #include <cmath> struct Color { int r, g, b; // Constructor
Color(int red, int green, int blue) : r(red), g(green), b(blue) {} }; double
calculateColorDistance(const Color& c1, const Color& c2) { // Euclidean distance
between two colors in RGB space return sqrt(pow(c1.r - c2.r, 2) + pow(c1.g -
c2.g, 2) + pow(c1.b - c2.b, 2)); } Color generateDistinctColor(const Color&
baseColor, double minDistance) { // Generate a color with a minimum distance
from the base color Color newColor = baseColor; // Randomly adjust the RGB
values newColor.r = (baseColor.r + rand() % 256) % 256; newColor.g =
(baseColor.g + rand() % 256) % 256; newColor.b = (baseColor.b + rand() % 256) %
256; // Ensure the new color has the minimum distance from the base color while
(calculateColorDistance(baseColor, newColor) < minDistance) { newColor.r =
(newColor.r + rand() % 256) % 256; newColor.g = (newColor.g + rand() % 256) %
256; newColor.b = (newColor.b + rand() % 256) % 256; } return newColor; } int
main() { // Example: Generate a palette of 5 distinct colors const int numColors
= 5; const double minColorDistance = 100.0; Color palette[numColors]; // Seed
the random number generator srand(static_cast<unsigned int>(time(NULL))); //
Generate the first color randomly palette[0] = Color(rand() % 256, rand() % 256,
rand() % 256); // Generate the rest of the colors with a minimum distance from
previous colors for (int i = 1; i < numColors; ++i) { palette[i] =
generateDistinctColor(palette[i - 1], minColorDistance); } // Display the
generated palette for (int i = 0; i < numColors; ++i) { std::cout << "Color " <<
i + 1 << ": RGB(" << palette[i].r << ", " << palette[i].g << ", " <<
palette[i].b << ")\n"; } return 0; }
```

This example uses the RGB color space and generates a palette of distinct colors with a specified minimum distance between them. Adjust the **`numColors`** and **`minColorDistance`** variables according to your requirements.