

Write a c++ program that does the following:


Provided a base color, create a list of five more colors that do not have problematic color combinations.



Certainly! To achieve this, you can use the HSL (Hue, Saturation, Lightness) color model in C++.

Here's a simple C++ program that takes a base color and generates five additional colors with no problematic color combinations:

cpp

 Copy code

```
#include <iostream> #include <cmath> struct RGBColor { int red; int green; int blue; }; struct HSLColor { double hue; double saturation; double lightness; }; HSLColor rgbToHsl(const RGBColor& rgb) { double r = rgb.red / 255.0; double g = rgb.green / 255.0; double b = rgb.blue / 255.0; double maxColor = std::max({r, g, b}); double minColor = std::min({r, g, b}); double delta = maxColor - minColor; double hue = 0.0; if (delta != 0.0) { if (maxColor == r) { hue = 60.0 * fmod((g - b) / delta, 6.0); } else if (maxColor == g) { hue = 60.0 * ((b - r) / delta + 2.0); } else if (maxColor == b) { hue = 60.0 * ((r - g) / delta + 4.0); } } double saturation = (maxColor == 0.0) ? 0.0 : (delta / maxColor); double lightness = (maxColor + minColor) / 2.0; return {hue, saturation, lightness}; } RGBColor hslToRgb(const HSLColor& hsl) { double hue = hsl.hue; double saturation = hsl.saturation; double lightness = hsl.lightness; double c = (1.0 - std::abs(2.0 * lightness - 1.0)) * saturation; double x = c * (1.0 - std::abs(fmod(hue / 60.0, 2.0) - 1.0)); double m = lightness - c / 2.0; double r, g, b; if (hue >= 0.0 && hue < 60.0) { r = c; g = x; b = 0.0; } else if (hue >= 60.0 && hue < 120.0) { r = x; g = c; b = 0.0; } else if (hue >= 120.0 && hue < 180.0) { r = 0.0; g = c; b = x; } else if (hue >= 180.0 && hue < 240.0) { r = 0.0; g = x; b = c; } else if (hue >= 240.0 && hue < 300.0) { r = x; g = 0.0; b = c; } else { r = c; g = 0.0; b = x; } return {static_cast<int>((r + m) * 255), static_cast<int>((g + m) * 255), static_cast<int>((b + m) * 255)}; } int main() { // Input base color in RGB format RGBColor baseColor = {255, 0, 0}; // Convert RGB to HSL HSLColor baseHSL = rgbToHsl(baseColor); // Generate five additional colors with no problematic combinations for (int i = 1; i <= 5; ++i) { // Increment hue for each additional color double newHue = fmod(baseHSL.hue + i * 30.0, 360.0); HSLColor newHSL = {newHue, baseHSL.saturation, baseHSL.lightness}; // Convert HSL to RGB RGBColor newColor = hslToRgb(newHSL); // Output the new color in RGB format std::cout << "Color " << i << ": (" << newColor.red << ", " << newColor.green << ", " << newColor.blue << ")\n"; } return 0; }
```

This program takes a base color in RGB format, converts it to HSL, increments the hue for five additional colors (30 degrees apart), and then converts them back to RGB. The resulting colors have no problematic combinations.