Question: Consider the following class definition whose objects are supposed to encapsulate data that represents a color. Each color is represented by its RGB (red, green, blue) content, where RGB values are integers from 0 to 255. For example, red color is represented as (255, 0, 0). Assume that when you add two colors, their corresponding RGB values get added but this addition is done modulo 256. For example:
- Adding (0, 0, 255) and (255, 0, 0) will result in (255, 0, 255).
- Adding (255, 255, 0) and (255, 1, 10) will give you (255, 0, 10).
- Adding (24, 43, 213) and (201, 222, 5) will give you (225, 9, 218)

```cpp
#include <iostream>
using namespace std;
class Color {
    int r, g, b;
public:
    Color(int r=0, int g=0, int b=0): r(r), g(g), b(b) {} 
}

friend ostream& operator<<(ostream& o, const Color& c); 
friend const Color operator+(const Color& left, const Color& right);
}
```

int main() {
    Color c1(111, 2, 250);
    Color c2(7, 103, 42);
    Color c3 = c1 + c2; // something required here
    cout << "Color1:"
    cout << c1 << endl;  // something required here
    cout << "Color2:"
    cout << c2 << endl;  // something required here
    cout << "Color3:"
    cout << c3 << endl;  // something required here
    return 0;
}

The sample output is:

<table>
<thead>
<tr>
<th>Color1:111,2,250</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color2:7,103,42</td>
</tr>
<tr>
<td>Color3:118,105,36</td>
</tr>
</tbody>
</table>

Provide whatever is required to make the program run and behave as expected. Provide declarations above and definitions on the next page.
ostream& operator<<(ostream& o, const Color& c) {
    return (o << c.r << "","" << c.g << "","" << c.b);
}

const Color operator+(const Color& left, const Color& right) {
    Color result;
    result.r = (left.r + right.r) % 256;
    result.g = (left.g + right.g) % 256;
    result.b = (left.b + right.b) % 256;
    return result;
}