```
#include <cstdint>
#include <iostream>

struct ip_v4
{
      uint8_t octet_1, octet_2, octet_3, octet_4;
      constexpr ip_v4() : octet_1(0), octet_2(0), octet_3(0), octet_4(0) {}
      constexpr ip_v4(uint8_t o1, uint8_t o2, uint8_t o3, uint8_t o4) :
octet_1(o1), octet_2(o2), octet_3(o3), octet_4(o4) {}
      constexpr ip_v4(uint32_t oct) : octet_1((oct >> 24) & 0xFF), octet_2((oct >>
16) & 0xFF), octet_3((oct >> 8) & 0xFF), octet_4(oct & 0xFF) {}
      uint32_t to_uint32() const { return (octet_1 << 24) | (octet_2 << 16) |
(octet_3 << 8) | octet_4; }
      ip_v4 operator&(const ip_v4 &other) const { return ip_v4(octet_1 &
other.octet_1, octet_2 & other.octet_2, octet_3 & other.octet_3, octet_4 &
other.octet_4); }
      ip_v4 operator|(const ip_v4 &other) const { return ip_v4(octet_1 |
other.octet_1, octet_2 | other.octet_2, octet_3 | other.octet_3, octet_4 |
other.octet_4); }
      ip_v4 operator~() const { return ip_v4(0xFF - octet_1, 0xFF - octet_2, 0xFF -
octet_3, 0xFF - octet_4); }
      ip_v4 operator^(const ip_v4 &other) const { return ip_v4(octet_1 ^
other.octet_1, octet_2 ^ other.octet_2, octet_3 ^ other.octet_3, octet_4 ^
other.octet_4); }
      bool operator==(const ip_v4 &other) const { return octet_1 == other.octet_1
&& octet_2 == other.octet_2 && octet_3 == other.octet_3 && octet_4 ==
other.octet_4; }
};

std::ostream &operator<<(std::ostream &stream, ip_v4 &ip) { return stream <<
std::to_string(ip.octet_1) << '.' << std::to_string(ip.octet_2) << '.' <<
std::to_string(ip.octet_3) << '.' << std::to_string(ip.octet_4); }
ip_v4 &operator>>(std::istream &stream, ip_v4 &ip)
{
      char c;
      int octet_1, octet_2, octet_3, octet_4;
      stream >> octet_1 >> c >> octet_2 >> c >> octet_3 >> c >> octet_4;
      ip = ip_v4(octet_1, octet_2, octet_3, octet_4);
      return ip;
}

constexpr ip_v4 class_a_mask = ip_v4(0xFF000000);
constexpr ip_v4 class_c_mask = ip_v4(0xFFFFFF00);
constexpr ip_v4 class_b_mask = ip_v4(0xFFFF0000);

int main()
{

      std::cout << "Enter a IP Address: ";
      ip_v4 ip;
      std::cin >> ip;
```

```
        std::cout << std::boolalpha;
        const ip_v4 masked_a = ip & class_a_mask;
        const ip_v4 masked_b = ip & class_b_mask;
        const ip_v4 masked_c = ip & class_b_mask;
        if (masked_a == ip_v4(10, 0, 0, 0))
        {
              std::cout << "Class A IP Address\n";
        }
        else if (masked_b == ip_v4(172, 16, 0, 0))
        {
              std::cout << "Class B IP Address\n";
        }
        else if (masked_c == ip_v4(192, 168, 0, 0))
        {
              std::cout << "Class C IP Address\n";
        }
        else
        {
              std::cout << "Unknown IP Address\n";
        }

        return 0;
}
```