**Bansilal Ramnath Agarwal Charitable Trust’s**



Vishwakarma Institute of Technology,Pune-37

*(An Autonomous Institute of Savitribai Phule Pune University)*

**Department of Artificial Intelligence and Data Science**

|  |  |
| --- | --- |
| **Division** | A |
| **Batch** | 03 |
| **GR-no** | 12311951 |
| **Rollno** | 25 |
| **Name** | Archit Bagad |

LAB ASSIGNMENT- 8

Code:-

#include <iostream>

#include <sstream>

#include <vector>

#include <cmath>

using namespace std;

// Function to calculate subnet mask

void calculateSubnetMask(int subnetBits, int subnetMask[]) {

    for (int i = 0; i < 4; i++) {

        if (subnetBits >= 8) {

            subnetMask[i] = 255;

            subnetBits -= 8;

        } else if (subnetBits > 0) {

            subnetMask[i] = 256 - (1 << (8 - subnetBits));

            subnetBits = 0;

        } else {

            subnetMask[i] = 0;

        }

    }

}

// Function to print an IP address

void printIP(const int ip[]) {

    for (int i = 0; i < 4; i++) {

        cout << ip[i];

        if (i < 3) cout << ".";

    }

    cout << endl;

}

// Function to determine the class of an IP address

char getIPClass(int firstOctet) {

    if (firstOctet >= 1 && firstOctet <= 126) return 'A';

    if (firstOctet >= 128 && firstOctet <= 191) return 'B';

    if (firstOctet >= 192 && firstOctet <= 223) return 'C';

    if (firstOctet >= 224 && firstOctet <= 239) return 'D'; // Multicast

    if (firstOctet >= 240 && firstOctet <= 255) return 'E'; // Experimental

    return 'X'; // Invalid

}

// Function to get the default subnet mask for an IP class

void getDefaultSubnetMask(char ipClass, int subnetMask[]) {

    if (ipClass == 'A') {

        subnetMask[0] = 255; subnetMask[1] = 0; subnetMask[2] = 0; subnetMask[3] = 0;

    } else if (ipClass == 'B') {

        subnetMask[0] = 255; subnetMask[1] = 255; subnetMask[2] = 0; subnetMask[3] = 0;

    } else if (ipClass == 'C') {

        subnetMask[0] = 255; subnetMask[1] = 255; subnetMask[2] = 255; subnetMask[3] = 0;

    } else {

        subnetMask[0] = subnetMask[1] = subnetMask[2] = subnetMask[3] = 0; // Default for Class D & E

    }

}

// Function to validate IP address format

bool isValidIP(const string &ip, int ipArr[]) {

    stringstream ss(ip);

    string segment;

    vector<int> segments;

    while (getline(ss, segment, '.')) {

        try {

            int num = stoi(segment);

            if (num < 0 || num > 255) return false;

            segments.push\_back(num);

        } catch (...) {

            return false;

        }

    }

    if (segments.size() != 4) return false;

    for (int i = 0; i < 4; i++) ipArr[i] = segments[i];

    return true;

}

int main() {

    string ipAddress;

    int subnetBits, ip[4], subnetMask[4];

    char ipClass;

    cout << "Enter IP address: ";

    cin >> ipAddress;

    if (!isValidIP(ipAddress, ip)) {

        cout << "Invalid IP address format!" << endl;

        return 1;

    }

    ipClass = getIPClass(ip[0]);

    if (ipClass == 'X') {

        cout << "Invalid or unsupported IP class!" << endl;

        return 1;

    }

    getDefaultSubnetMask(ipClass, subnetMask);

    cout << "IP Class: " << ipClass << endl;

    cout << "Default Subnet Mask: ";

    printIP(subnetMask);

    cout << "Enter the number of subnet bits (0-30): ";

    cin >> subnetBits;

    if (subnetBits < 0 || subnetBits > 30) {

        cout << "Invalid subnet bits! Must be between 0 and 30." << endl;

        return 1;

    }

    calculateSubnetMask(subnetBits, subnetMask);

    cout << "IP Address: ";

    printIP(ip);

    cout << "Calculated Subnet Mask: ";

    printIP(subnetMask);

    return 0;

}

Output:

