Socket java or python programming

#include <iostream>

#include <string>

using namespace std;

bool isValidIP(const string& ip) {

int octetCount = 0;

int currentOctet = 0;

bool isDigit = false;

for (char c : ip) {

if (c == '.') {

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

currentOctet = 0;

++octetCount;

isDigit = false;

} else if (isdigit(c)) {

currentOctet = currentOctet \* 10 + (c - '0');

isDigit = true;

} else {

return false;

}

}

if (!isDigit || currentOctet > 255 || octetCount != 3) return false;

return true;

}

void classless() {

}

void classfull() {

string ip;

cout << "Enter IP Address: ";

cin >> ip;

if (!isValidIP(ip)) {

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

return;

}

int firstOctet = 0;

size\_t pos = ip.find('.');

if (pos != string::npos) {

firstOctet = 0;

for (size\_t i = 0; i < pos; ++i) {

firstOctet = firstOctet \* 10 + (ip[i] - '0');

}

}

string nwclass;

string mask;

if (firstOctet >= 1 && firstOctet <= 126) {

nwclass = "A";

mask = "255.0.0.0";

}

else if (firstOctet >= 128 && firstOctet <= 191) {

nwclass = "B";

mask = "255.255.0.0";

}

else if (firstOctet >= 192 && firstOctet <= 223) {

nwclass = "C";

mask = "255.255.255.0";

}

else {

nwclass = "Reserved";

mask = "-";

}

cout << "IP Address: " << ip << endl;

cout << "Class: " << nwclass << endl;

cout << "Default Mask: " << mask << endl;

}

int main() {

int choice;

cout << "Menu:\n";

cout << "1. Classful Addressing" << endl;

cout << "2. Classless Addressing" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Classful Addressing" << endl;

classfull();

break;

case 2:

cout << "Classless Addressing" << endl;

classless();

break;

default:

cout << "Invalid choice." << endl;

break;

}

return 0;

}  
  
  
  
  
IP Subnetting:  
#include <bits/stdc++.h>

using namespace std;

int main(void)

{

cout << "Enter IP in form a.b.c.d\n";

vector<int> ip(4);

cout << "a: ";

cin >> ip[0];

cout << "b: ";

cin >> ip[1];

cout << "c: ";

cin >> ip[2];

cout << "d: " ;

cin >> ip[3];

int mask;

if(ip[0] <= 127)mask = 0;

else if(ip[0] <= 191)mask = 1;

else if(ip[0] <= 223)mask = 2;

else if(ip[0] <= 255)mask = 3;

vector<int> masks(4);

if(mask >= 0)masks[0]=255;

if(mask >= 1)masks[1] = 255;

if(mask >= 2)masks[2] = 255;

cout << "Enter the number of subnets" << endl;

int n; cin >> n;

int num = 1;

while(num < n)num \*= 2;

int bits = log2(num);

int tot = (mask + 1)\*8+bits;

int copy = 32-tot;

// cout << tot << endl;

vector<int> subnet\_mask(4);

for(int i = 0; i < 4; i++)

{

int a = 0;

for(int j = 7; j >= 0; j--)

{

if(tot > 0)

{

a = (a | (1 << j));

tot--;

}

}

subnet\_mask[i] = a;

}

cout << "The IP is: " << ip[0]<<"."<<ip[1]<<"."<<ip[2]<<"."<<ip[3] << endl;

cout << "The Class is: " << (char)('A'+mask) << endl;

cout << "The subnet mask is: " << subnet\_mask[0]<<"."<<subnet\_mask[1]<<"."<<subnet\_mask[2]<<"."<<subnet\_mask[3] << endl;

cout << "The IP Addresses are: " << endl;

int amount = pow(2,copy);

cout << copy << amount << endl;

for(int i = 0; i < num; i++)

{

cout << "Block " << i + 1 << endl;

cout << "Start: " << ip[0]<<"."<<ip[1]<<"."<<ip[2]<<"."<<ip[3] << endl;

ip[3] += amount - 1;

for(int j = 3; j >= 1; j --)

{

while(ip[j] > 255)

{

ip[j]-=256;

ip[j-1]+=1;

}

}

cout << "End: " << ip[0]<<"."<<ip[1]<<"."<<ip[2]<<"."<<ip[3] << endl;

ip[3]++;

for(int j = 3; j >= 1; j --)

{

while(ip[j] > 255)

{

ip[j]-=256;

ip[j-1]+=1;

}

}

}

}

Final code:

#include <bits/stdc++.h>

using namespace std;

bool isValidIP(const string& ip) {

int octetCount = 0;

int currentOctet = 0;

bool isDigit = false;

for (char c : ip) {

if (c == '.') {

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

currentOctet = 0;

++octetCount;

isDigit = false;

} else if (isdigit(c)) {

currentOctet = currentOctet \* 10 + (c - '0');

isDigit = true;

} else {

return false;

}

}

if (!isDigit || currentOctet > 255 || octetCount != 3) return false;

return true;

}

bool isValidCIDR(const string& cidr) {

size\_t pos = cidr.find('/');

if (pos == string::npos) return false;

string ipPart = cidr.substr(0, pos);

string maskPart = cidr.substr(pos + 1);

return isValidIP(ipPart) && !maskPart.empty() && stoi(maskPart) >= 0 && stoi(maskPart) <= 32;

}

void classless() {

string cidr;

cout << "Enter IP Address in CIDR notation (e.g., 192.168.1.0/24): ";

cin >> cidr;

if (!isValidCIDR(cidr)) {

cout << "Invalid CIDR format." << endl;

return;

}

size\_t pos = cidr.find('/');

string ip = cidr.substr(0, pos);

int prefixLength = stoi(cidr.substr(pos + 1));

vector<int> subnetMask(4);

for (int i = 0; i < 4; ++i) {

if (prefixLength >= 8) {

subnetMask[i] = 255;

prefixLength -= 8;

} else {

subnetMask[i] = (1 << prefixLength) - 1;

subnetMask[i] <<= (8 - prefixLength);

break;

}

}

cout << "IP Address: " << ip << endl;

cout << "Subnet Mask: " << subnetMask[0] << "." << subnetMask[1] << "." << subnetMask[2] << "." << subnetMask[3] << endl;

}

void classfull() {

string ip;

cout << "Enter IP Address: ";

cin >> ip;

if (!isValidIP(ip)) {

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

return;

}

int firstOctet = stoi(ip.substr(0, ip.find('.')));

string nwclass;

vector<int> mask(4, 0);

if (firstOctet >= 1 && firstOctet <= 126) {

nwclass = "A";

mask = {255, 0, 0, 0};

} else if (firstOctet >= 128 && firstOctet <= 191) {

nwclass = "B";

mask = {255, 255, 0, 0};

} else if (firstOctet >= 192 && firstOctet <= 223) {

nwclass = "C";

mask = {255, 255, 255, 0};

} else {

nwclass = "Reserved";

}

cout << "IP Address: " << ip << endl;

cout << "Class: " << nwclass << endl;

cout << "Default Mask: " << mask[0] << "." << mask[1] << "." << mask[2] << "." << mask[3] << endl;

}

void subnetting() {

vector<int> ip(4);

cout << "Enter IP in form a.b.c.d:\n";

cout << "a: "; cin >> ip[0];

cout << "b: "; cin >> ip[1];

cout << "c: "; cin >> ip[2];

cout << "d: "; cin >> ip[3];

int mask;

if (ip[0] <= 127) mask = 0;

else if (ip[0] <= 191) mask = 1;

else if (ip[0] <= 223) mask = 2;

else mask = 3;

vector<int> subnetMask(4);

if (mask >= 0) subnetMask[0] = 255;

if (mask >= 1) subnetMask[1] = 255;

if (mask >= 2) subnetMask[2] = 255;

cout << "Enter the number of subnets: ";

int n; cin >> n;

int num = 1;

while (num < n) num \*= 2;

int bits = log2(num);

int totalBits = (mask + 1) \* 8 + bits;

int copy = 32 - totalBits;

vector<int> newSubnetMask(4);

for (int i = 0; i < 4; i++) {

int a = 0;

for (int j = 7; j >= 0; j--) {

if (totalBits > 0) {

a = (a | (1 << j));

totalBits--;

}

}

newSubnetMask[i] = a;

}

cout << "The IP is: " << ip[0] << "." << ip[1] << "." << ip[2] << "." << ip[3] << endl;

cout << "The Class is: " << (char)('A' + mask) << endl;

cout << "The subnet mask is: " << newSubnetMask[0] << "." << newSubnetMask[1] << "." << newSubnetMask[2] << "." << newSubnetMask[3] << endl;

int amount = pow(2, copy);

for (int i = 0; i < num; i++) {

cout << endl;

cout << "Subnet " << i + 1 << endl;

cout << "1st valid address: " << ip[0] << "." << ip[1] << "." << ip[2] << "." << ip[3] << endl;

ip[3] += amount - 1;

for (int j = 3; j >= 1; j--) {

while (ip[j] > 255) {

ip[j] -= 256;

ip[j - 1] += 1;

}

}

cout << "Last valid address: " << ip[0] << "." << ip[1] << "." << ip[2] << "." << ip[3] << endl;

ip[3]++;

for (int j = 3; j >= 1; j--) {

while (ip[j] > 255) {

ip[j] -= 256;

ip[j - 1] += 1;

}

}

}

}

int main() {

int choice;

while (true) {

cout << "\nMenu:\n";

cout << "1. Classful Addressing" << endl;

cout << "2. Classless Addressing" << endl;

cout << "3. IP Subnetting" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

cout << endl;

switch (choice) {

case 1:

cout << "Classful Addressing" << endl;

classfull();

break;

case 2:

cout << "Classless Addressing" << endl;

classless();

break;

case 3:

cout << "IP Subnetting" << endl;

subnetting();

break;

case 4:

cout << "Exit" << endl;

return 0;

default:

cout << "Invalid choice. Please try again." << endl;

break;

}

}

return 0;

}

  
  
  
  
  
  
