



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE 1004

Network and Communication

LAB ASSESSMENT - 3

NAME: Vibhu Kumar Singh

REG. NO: 19BCE0215

TEACHER: Santhi H.

1. Write a menu driven code for Decimal, Binary

- To check the class, network id and host id of an IPv4 address.
(Use function wherever necessary)
- To check whether given IP address is valid or not.
- To find first address, last address and number of addresses in the block.

Ans 1)

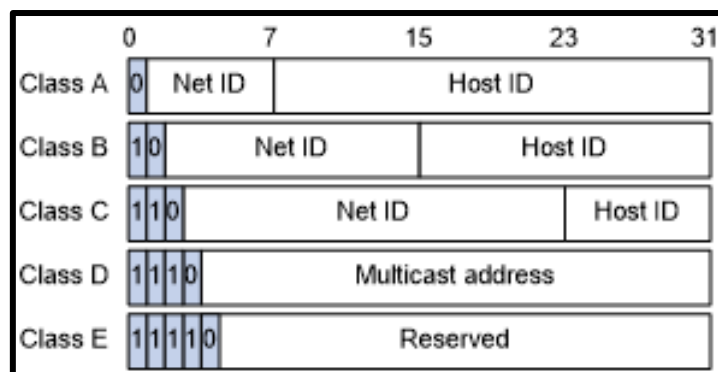
Aim: To check for Validity, Class, Network ID, Host ID, First Address, Last Address and Number of Addresses of a user input IPv4 Address (classful).

Algorithms:

- Validity:

Each Byte of an IPv4 Address must be between 0 and 255 (0-11111111 if binary) and there should be exactly 4 such bytes.

- Class:



Class A : First Byte must be between 0 and 127 or start with 0 if binary input.

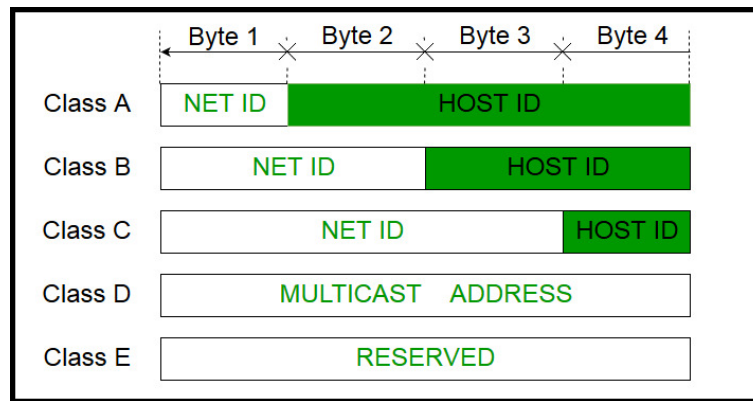
Class B : First Byte must be between 128 and 191 or start with 10 if binary input.

Class C : First Byte must be between 192 and 223 or start with 110 if binary input.

Class D : First Byte must be between 224 and 239 or start with 1110 if binary input.

Class E : First Byte must be between 240 and 255 or start with 11111 if binary input.

- Network ID/Host ID:



Class A : First Byte is the Network ID, rest is Host ID.

Class B : First and Second is Network ID, rest is Host ID.

Class C : First, second and third bytes is Network ID, rest is Host ID.

Class D : Multicast Addresses.

Class E : Reserved Addresses.

- First/Last Addresses:

Class A : First Address is 0.0.0.0

Last Address is 127.255.255.255

Class B : First Address is 128.0.0.0

Last Address is 191.255.255.255

Class C : First Address is 192.0.0.0

Last Address is 223.255.255.255

Class D : Invalid as there is no subnet.

Class E : Invalid as there is no subnet.

Menu-Driven Source Code:

```
#include<bits/stdc++.h>
using namespace std;

int DecimalOrBinary();
void DecimalIPClass();
bool ValidIP(string,int);
void DecimalFirstLast();
void BinaryIPClass();
int toDecimal(string);
void BinaryFirstLast();

void IPClassMenu()
{
    int decimalBinary=DecimalOrBinary();
    switch(decimalBinary)
    {
        case 1:
            DecimalIPClass();
            break;
        case 2:
            BinaryIPClass();
            break;
    }
    cout<<"\n-----\n\n\n";
}

void DecimalIPClass()
{
    start:
    string ip;
    cout<<"Enter the Decimal IP Address: ";
    cin>>ip;
    if(ValidIP(ip,1))
    {
        string block1="";
        string block2="";
        string block3="";
        string block4="";
        int i=0;
        while(ip[i]!='.')
        {
            block1.push_back(ip[i]);
            i++;
        }
        i++;
        while(ip[i]!='.')
        {
            block2.push_back(ip[i]);
            i++;
        }
        i++;
        while(ip[i]!='.')
        {
            block3.push_back(ip[i]);
            i++;
        }
        i++;
    }
}
```

```

        while(ip[i])
        {
            block4.push_back(ip[i]);
            i++;
        }
        if(stoi(block1)<=127)
        {
            cout<<"\nClass: Class A\nNetwork ID: "<<block1<<"\nHost ID: "<<block2<<". "<<block3<<". "<<block4;
        }
        else if(stoi(block1)>=128 && stoi(block1)<=191)
        {
            cout<<"\nClass: Class B\nNetwork ID: "<<block1<<". "<<block2<<"\nHost ID: "<<block3<<". "<<block4;
        }
        else if(stoi(block1)>=192 && stoi(block1)<=223)
        {
            cout<<"\nClass: Class C\nNetwork ID: "<<block1<<". "<<block2<<". "<<block3<<"\nHost ID: "<<block4;
        }
        else if(stoi(block1)>=224 && stoi(block1)<=239)
        {
            cout<<"\nClass: Class D";
        }
        else
        {
            cout<<"\nClass: Class E";
        }
    }
    else
    {
        cout<<"\nEnter a valid IP Address.\n";
        goto start;
    }
}

void BinaryIPClass()
{
    start:
    string ip;
    cout<<"Enter the Binary IP Address: ";
    cin>>ip;
    if(ValidIP(ip,2))
    {
        string block1="";
        string block2="";
        string block3="";
        string block4="";
        int i=0;
        while(ip[i]!='.')
        {
            block1.push_back(ip[i]);
            i++;
        }
        i++;
        while(ip[i]!='.')
        {
            block2.push_back(ip[i]);
            i++;
        }
    }
}

```

```

        i++;
        while(ip[i]!='.')
        {
            block3.push_back(ip[i]);
            i++;
        }
        i++;
        while(ip[i])
        {
            block4.push_back(ip[i]);
            i++;
        }
        if(toDecimal(block1)<=127)
        {
            cout<<"\nClass: Class A\nNetwork ID: "<<block1<<"\nHost ID: "<<block2<<"."<<block3<<"."<<block4;
        }
        else if(toDecimal(block1)>=128 && toDecimal(block1)<=191)
        {
            cout<<"\nClass: Class B\nNetwork ID: "<<block1<<"."<<block2<<"\nHost ID: "<<block3<<"."<<block4;
        }
        else if(toDecimal(block1)>=192 && toDecimal(block1)<=223)
        {
            cout<<"\nClass: Class C\nNetwork ID: "<<block1<<"."<<block2<<"."<<block3<<"\nHost ID: "<<block4;
        }
        else if(toDecimal(block1)>=224 && toDecimal(block1)<=239)
        {
            cout<<"\nClass: Class D is for Multicast Addresses";
        }
        else
        {
            cout<<"\nClass: Class E is for Reserved Addresses";
        }
    }
    else
    {
        cout<<"\nEnter a valid IP Address.\n";
        goto start;
    }
}

int toDecimal(string s)
{
    return stoi(s,0,2);
}

bool ValidIP(string ip, int decimalBinary)
{
    int j=0;
    int dots=0;
    while(ip[j])
    {
        if(!isdigit(ip[j]) && ip[j]!='.')
            return false;
        if(ip[j]=='.')
            dots++;
        j++;
    }
}

```

```

string block1="";
string block2="";
string block3="";
string block4="";
int i=0;
while(ip[i]!='.' && i<ip.length())
{
    block1.push_back(ip[i]);
    i++;
}
i++;
while(ip[i]!='.' && i<ip.length())
{
    block2.push_back(ip[i]);
    i++;
}
i++;
while(ip[i]!='.' && i<ip.length())
{
    block3.push_back(ip[i]);
    i++;
}
i++;
while(ip[i])
{
    block4.push_back(ip[i]);
    i++;
}
int flag=0;
if(decimalBinary==1)
{
    if(dots==3 && !block1.empty() && !block2.empty() && !block3.empty() && !block4.empty())
    {
        if(stoi(block1)>=0 && stoi(block1)<=255)
        {
            if(stoi(block2)>=0 && stoi(block2)<=255)
            {
                if(stoi(block3)>=0 && stoi(block3)<=255)
                {
                    if(stoi(block4)>=0 && stoi(block4)<=255)
                    {
                        flag=1;
                        return true;
                    }
                }
            }
        }
    }
    if(flag==0)
    {
        return false;
    }
}
else if(decimalBinary==2)
{
    int k=0;
    while(ip[k])
    {
        if(ip[k]!='1' && ip[k]!='0' && ip[k]!='.')

```

```

        {
            return false;
        }
        k++;
    }
    if(dots==3 && !block1.empty() && !block2.empty() && !block3.empty() && !block4.empty())
    {
        if(toDecimal(block1)>=0 && toDecimal(block1)<=255)
        {
            if(toDecimal(block2)>=0 && toDecimal(block2)<=255)
            {
                if(toDecimal(block3)>=0 && toDecimal(block3)<=255)
                {
                    if(toDecimal(block4)>=0 && toDecimal(block4)<=255)
                    {
                        return true;
                    }
                }
            }
        }
    }
    else
    {
        return false;
    }
}
}

```

```

int DecimalOrBinary()
{
    start:
    int choice;
    cout<<"1.Decimal\n2.Binary\nEnter your choice: ";
    cin>>choice;
    cout<<"-----\n\n";
    switch(choice)
    {
        case 1:
            return 1;
        case 2:
            return 2;
        default:
            cout<<"\nInvalid Choice\n";
            goto start;
    }
}

```

```

void ValidIPMenu()
{
    int decimalBinary=DecimalOrBinary();
    if(decimalBinary==1)
    {
        cout<<"Enter Decimal IP Address: ";
    }
    if(decimalBinary==2)
    {
        cout<<"Enter Binary IP Address: ";
    }
    string ip;
}

```



```

cin>>ip;
bool isvalid=ValidIP(ip,decimalBinary);
if(isvalid)
{
    cout<<"\nThe input IP Address is VALID.\n";
}
else
{
    cout<<"\nThe input IP Address is INVALID\n";
}
cout<<"\n-----\n\n\n";
}

```

```

void FirstLastMenu()
{
    int decimalBinary=DecimalOrBinary();
    switch(decimalBinary)
    {
        case 1:
            DecimalFirstLast();
            break;
        case 2:
            BinaryFirstLast();
            break;
    }
    cout<<"\n-----\n\n\n";
}

```

```

void DecimalFirstLast()
{
    start:
    string ip;
    cout<<"\nEnter the Decimal IP Address: ";
    cin>>ip;
    if(ValidIP(ip,1))
    {
        string block1="";
        string block2="";
        string block3="";
        string block4="";
        int i=0;
        while(ip[i]!='.')
        {
            block1.push_back(ip[i]);
            i++;
        }
        i++;
        while(ip[i]!='.')
        {
            block2.push_back(ip[i]);
            i++;
        }
        i++;
        while(ip[i]!='.')
        {
            block3.push_back(ip[i]);
            i++;
        }
        i++;
        while(ip[i])

```

```

        {
            block4.push_back(ip[i]);
            i++;
        }
        if(stoi(block1)<=127)
        {
            cout<<"First Address: "<<block1<<".0.0.0\nLast Address: "<<block1<<".255.255.2
55\nNumber of Addresses: "<<(long long int)pow(2,24);
        }
        else if(stoi(block1)>=128 && stoi(block1)<=191)
        {
            cout<<"First Address: "<<block1<<"."<<block2<<".0.0\nLast Address: "<<block1<<
".255.255\nNumber of Addresses: "<<(long long int)pow(2,16);
        }
        else if(stoi(block1)>=192 && stoi(block1)<=223)
        {
            cout<<"First Address: "<<block1<<"."<<block2<<block3<<".0\nLast Address: "<<bl
ock1<<".255\nNumber of Addresses: "<<pow(2,8);
        }
        else if(stoi(block1)>=224 && stoi(block1)<=239)
        {
            cout<<"First Address: Invalid\nLast Address: Invalid\nNumber of Addresses: Inv
alid";
        }
        else if(stoi(block1)>=240 && stoi(block1)<=255)
        {
            cout<<"First Address: Invalid\nLast Address: Invalid\nNumber of Addresses: Inv
alid";
        }
    }
    else
    {
        cout<<"\nEnter a valid IP Address.\n";
        goto start;
    }
}

void BinaryFirstLast()
{
    start:
    string ip;
    cout<<"\nEnter the Decimal IP Address: ";
    cin>>ip;
    if(ValidIP(ip,2))
    {
        string block1="";
        string block2="";
        string block3="";
        string block4="";
        int i=0;
        while(ip[i]!='.')
        {
            block1.push_back(ip[i]);
            i++;
        }
        i++;
        while(ip[i]!='.')
        {
            block2.push_back(ip[i]);
            i++;
        }
    }
}

```

```

    }
    i++;
    while(ip[i]!='.')
    {
        block3.push_back(ip[i]);
        i++;
    }
    i++;
    while(ip[i])
    {
        block4.push_back(ip[i]);
        i++;
    }
    if(toDecimal(block1)<=127)
    {
        cout<<"First Address: "<<block1<<".0.0.0\nLast Address: "<<block1<<".11111111.
11111111.11111111\nNumber of Addresses: "<<(long long int)pow(2,24);
    }
    else if(toDecimal(block1)>=128 && toDecimal(block1)<=191)
    {
        cout<<"First Address: "<<block1<<". "<<block2<<".0.0\nLast Address: "<<block1<<
".11111111.11111111\nNumber of Addresses: "<<(long long int)pow(2,16);
    }
    else if(toDecimal(block1)>=192 && toDecimal(block1)<=223)
    {
        cout<<"First Address: "<<block1<<". "<<block2<<block3<<".0\nLast Address: "<<bl
ock1<<".11111111\nNumber of Addresses: "<<pow(2,8);
    }
    else if(toDecimal(block1)>=224 && toDecimal(block1)<=239)
    {
        cout<<"First Address: Invalid\nLast Address: Invalid\nNumber of Addresses: Inv
alid";
    }
    else
    {
        cout<<"First Address: Invalid\nLast Address: Invalid\nNumber of Addresses: Inv
alid";
    }
}
else
{
    cout<<"\nEnter a valid IP Address.";
    goto start;
}
}

void options()
{
    while(true)
    {
        cout<<"-----IPv4 Addressing Main Menu-----
\n        by [VIBHU KUMAR SINGH]\n\n";
        cout<<"1.To check the class, network id and host id of an IPv4 address\n2.To check
whether given IP address is valid or not\n3.To find first address, last address and numbe
r of addresses in the block\n0. Exit\nEnter your choice: ";
        int choice;
        cin>>choice;
        cout<<"-----
-\n\n";
        switch(choice)

```

```

{
    case 1:
        IPClassMenu();
        break;
    case 2:
        ValidIPMenu();
        break;
    case 3:
        FirstLastMenu();
        break;
    case 0:
        exit(0);
        break;
    default:
        cout<<"\nInvalid Choice\n";
        break;
}
}
}

int main()
{
    system("cls");
    options();
    return 0;
}

```

OUTPUT SCREENSHOTS:

a) To check the class, network id and host id of an IPv4 address :

i. Decimal IP:



```

File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-----IPv4 Addressing Main Menu-----
by [VIBHU KUMAR SINGH]

1.To check the class, network id and host id of an IPv4 address
2.To check whether given IP address is valid or not
3.To find first address, last address and number of addresses in the block
0. Exit
Enter your choice: 1
-----

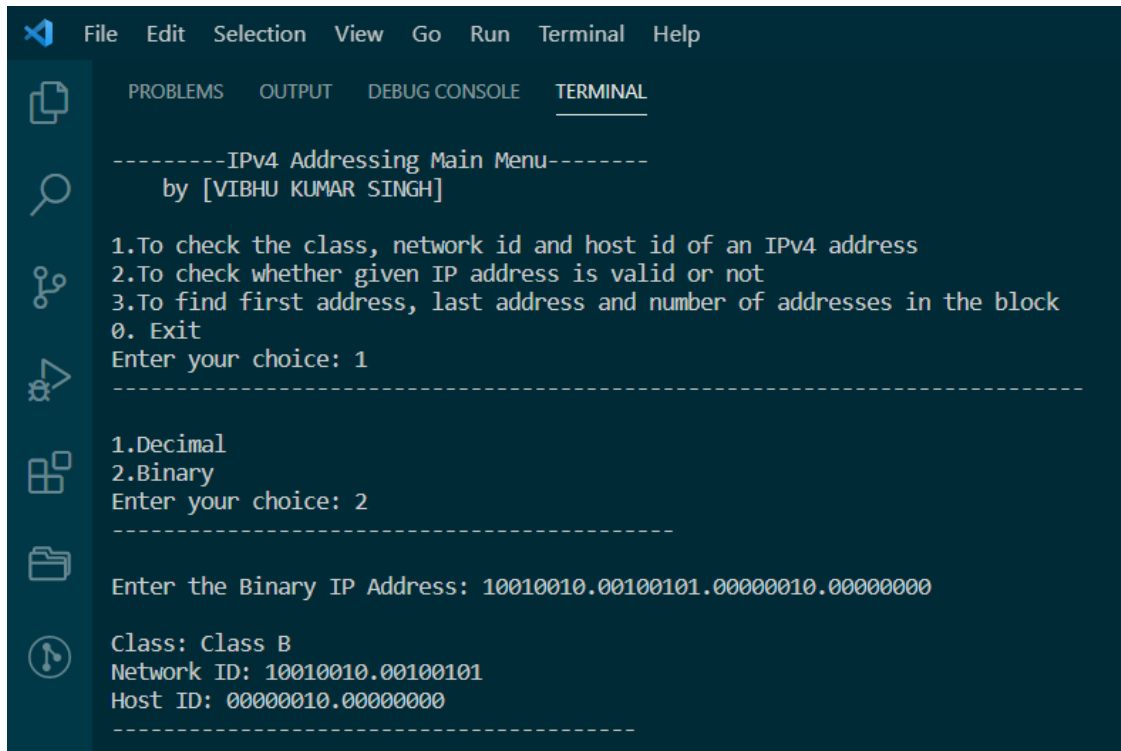
1.Decimal
2.Binary
Enter your choice: 1
-----

Enter the Decimal IP Address: 193.234.0.1

Class: Class C
Network ID: 193.234.0
Host ID: 1
-----

```

ii. Binary IP:



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-----IPv4 Addressing Main Menu-----
by [VIBHU KUMAR SINGH]

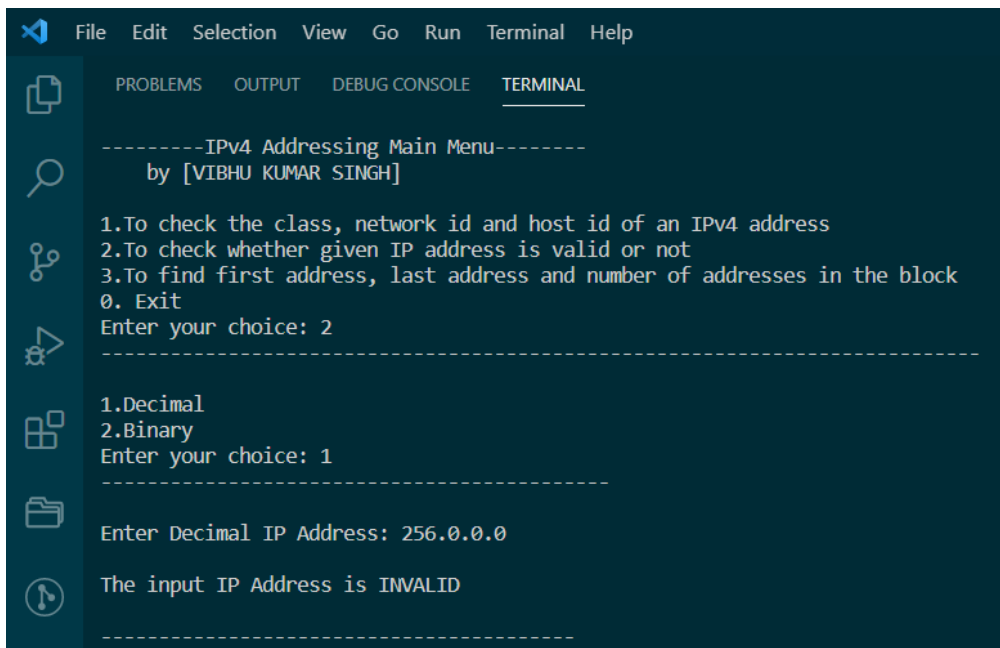
1.To check the class, network id and host id of an IPv4 address
2.To check whether given IP address is valid or not
3.To find first address, last address and number of addresses in the block
0. Exit
Enter your choice: 1
-----
1.Decimal
2.Binary
Enter your choice: 2
-----
Enter the Binary IP Address: 10010010.00100101.00000010.00000000

Class: Class B
Network ID: 10010010.00100101
Host ID: 00000010.00000000
-----
```

b) To check whether given IP address is valid or not :

i. Decimal IP:

- Invalid:



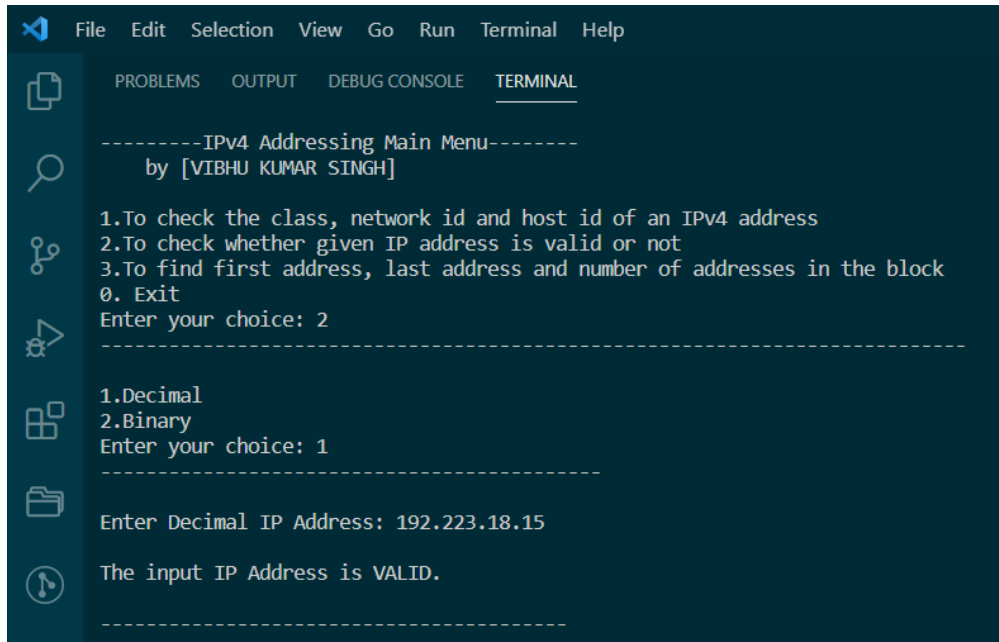
```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-----IPv4 Addressing Main Menu-----
by [VIBHU KUMAR SINGH]

1.To check the class, network id and host id of an IPv4 address
2.To check whether given IP address is valid or not
3.To find first address, last address and number of addresses in the block
0. Exit
Enter your choice: 2
-----
1.Decimal
2.Binary
Enter your choice: 1
-----
Enter Decimal IP Address: 256.0.0.0

The input IP Address is INVALID
-----
```

(Invalid because first byte is greater than 255)

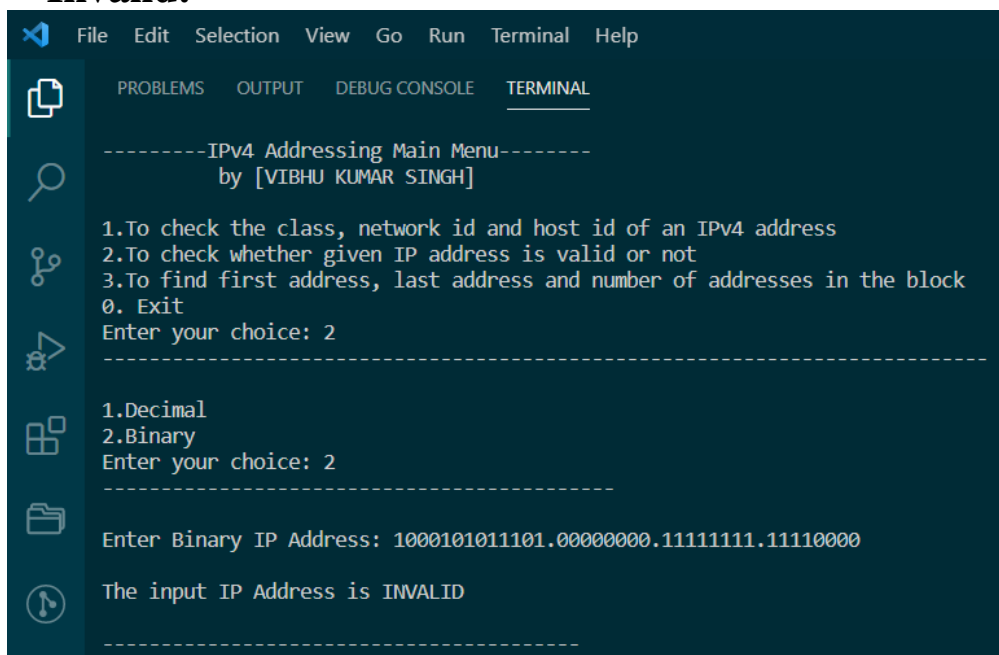
- **Valid:**



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-----IPv4 Addressing Main Menu-----
by [VIBHU KUMAR SINGH]
1.To check the class, network id and host id of an IPv4 address
2.To check whether given IP address is valid or not
3.To find first address, last address and number of addresses in the block
0. Exit
Enter your choice: 2
-----
1.Decimal
2.Binary
Enter your choice: 1
-----
Enter Decimal IP Address: 192.223.18.15
The input IP Address is VALID.
-----
```

ii. **Binary IP:**

- **Invalid:**



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-----IPv4 Addressing Main Menu-----
by [VIBHU KUMAR SINGH]
1.To check the class, network id and host id of an IPv4 address
2.To check whether given IP address is valid or not
3.To find first address, last address and number of addresses in the block
0. Exit
Enter your choice: 2
-----
1.Decimal
2.Binary
Enter your choice: 2
-----
Enter Binary IP Address: 1000101011101.00000000.11111111.11110000
The input IP Address is INVALID
-----
```

(invalid because first byte is greater than 8 bits)

- **Valid:**



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-----IPv4 Addressing Main Menu-----
by [VIBHU KUMAR SINGH]
1.To check the class, network id and host id of an IPv4 address
2.To check whether given IP address is valid or not
3.To find first address, last address and number of addresses in the block
0. Exit
Enter your choice: 2
-----
1.Decimal
2.Binary
Enter your choice: 2
-----
Enter Binary IP Address: 1001001.00101010.00010001.10000000
The input IP Address is VALID.
-----
```

c) **To find first address, last address and number of addresses in the block :**

i. **Decimal IP:**



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-----IPv4 Addressing Main Menu-----
by [VIBHU KUMAR SINGH]
1.To check the class, network id and host id of an IPv4 address
2.To check whether given IP address is valid or not
3.To find first address, last address and number of addresses in the block
0. Exit
Enter your choice: 3
-----
1.Decimal
2.Binary
Enter your choice: 1
-----
Enter the Decimal IP Address: 193.125.35.24
First Address: 193.12535.0
Last Address: 193.255
Number of Addresses: 256
-----
```

ii. Binary IP:

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

-----IPv4 Addressing Main Menu-----
by [VIBHU KUMAR SINGH]

1.To check the class, network id and host id of an IPv4 address
2.To check whether given IP address is valid or not
3.To find first address, last address and number of addresses in the block
0. Exit
Enter your choice: 3
-----
1.Decimal
2.Binary
Enter your choice: 2
-----
Enter the Decimal IP Address: 10001000.00110011.01010101.00001111
First Address: 10001000.00110011.0.0
Last Address: 10001000.11111111.11111111
Number of Addresses: 65536
-----
```

2. Implement the following unicast routing algorithms using functions.

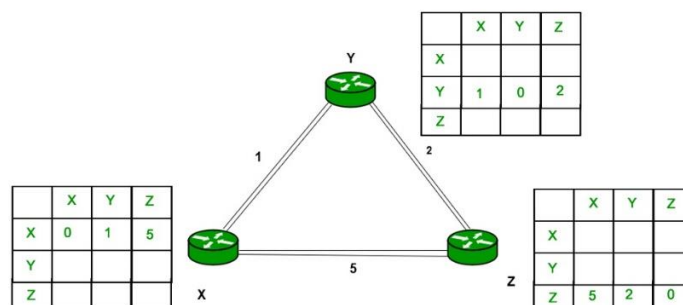
- a. Distance Vector Routing
- b. Link State Routing

Ans 2)

Aim: To implement Distance Vector and Link State Routing Techniques in C++.

Algorithms:

a) Distance Vector Routing:



1. A router transmits its distance vector to each of its neighbours in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbours.
3. A router recalculates its distance vector when:
 - It receives a distance vector from a neighbour containing different information than before.
 - It discovers that a link to a neighbour has gone down.

b) Link State Routing:

- **Step-1:** The node is taken and chosen as a root node of the tree, this creates the tree with a single node, and now set the total cost of each node to some value based on the information in Link State Database
- **Step-2:** Now the node selects one node, among all the nodes not in the tree like structure, which is nearest to the root, and adds this to the tree. The shape of the tree gets changed .
- **Step-3:** After this node is added to the tree, the cost of all the nodes not in the tree needs to be updated because the paths may have been changed.
- **Step-4:** The node repeats the Step 2. and Step 3. until all the nodes are added in the tree

Menu-Driven Source Code:

```
#include<bits/stdc++.h>
#define INF -1
#define MAX 10
using namespace std;

void LinkState();
void DistanceVector();

void DistanceVector()
{
    int graph[50][50];
    int i,j,k,t;
    int nn;
    cout<<"\n Enter Number of Nodes:";
    cin>>nn;
    for (i=0;i<nn;i++)
```

```

{
    for(j=0;j<nn;j++)
    {
        graph[i][j]=-1;
    }
}
char ch[7]={'A','B','C','D','E','F','G'};
for (i=0;i<nn;i++)
{
    for(j=0;j<nn;j++)
    {
        if(i==j)
        {
            graph[i][j]=0;
        }
        if(graph[i][j]==-1)
        {
            cout<<"\n Enter Distance between "<<ch[i]<<" - "<<ch[j]<<" : ";
            cin>>t;
            graph[i][j]=graph[j][i]=t;
        }
    }
}
int via[50][50];
for (i=0;i<nn;i++)
{
    for(j=0;j<nn;j++)
    {
        via[i][j]=-1;
    }
}
cout<<"\n After Initialization";
for (i=0;i<nn;i++)
{
    cout<<"\n"<<ch[i]<<" Table";
    cout<<"\nNode\tDist\tVia";
    for(j=0;j<nn;j++)
    {
        cout<<"\n"<<ch[j]<<"\t"<<graph[i][j]<<"\t"<<via[i][j];
    }
}
int sh[50][50][50];
for(i=0;i<nn;i++)
{
    for(j=0;j<nn;j++)
    {
        for (k=0;k<nn;k++)
        {
            if((graph[i][j]>-1)&&(graph[j][k]>-1))
            {
                sh[i][j][k]=graph[j][k]+graph[i][j];
            }
            else
            {
                sh[i][j][k]=-1;
            }
        }
    }
}
for(i=0;i<nn;i++)

```

```

{
    cout<<"\n\n For "<<ch[i];
    for (j=0;j<nn;j++)
    {
        cout<<"\n From "<<ch[j];
        for(k=0;k<nn;k++)
        {
            cout<<"\n "<<ch[k]<<" "<<sh[i][j][k];
        }
    }
}
int final[50][50];
for(i=0;i<nn;i++)
{
    for(j=0;j<nn;j++)
    {
        final[i][j]=graph[i][j];
        via[i][j]=i;
        for(k=0;k<nn;k++)
        {
            if((final[i][j]>sh[i][k][j]) || (final[i][j] == -1))
            {
                if(sh[i][k][j]>-1)
                {
                    final[i][j]=sh[i][k][j];
                    via[i][j]=k;
                }
            }
        }
        if(final[i][j]==-1)
        {
            for(k=0;k<nn;k++)
            {
                if((final[i][k]!=-1)&&(final[k][j]!=-1))
                {
                    if((final[i][j]==-1) || ((final[i][j]!=-
1) &&(final[i][j]>final[i][k]+final[k][j])))
                    {
                        if(final[i][k]+final[k][j]>-1)
                        {
                            final[i][j]=final[i][k]+final[k][j];
                            via[i][j]=k;
                        }
                    }
                }
            }
        }
    }
}
cout<<"\n After Update :";
for (i=0;i<nn;i++)
{
    cout<<"\n"<<ch[i]<<" Table";
    cout<<"\nNode\tDist\tVia";
    for(j=0;j<nn;j++)
    {
        cout<<"\n"<<ch[j]<<"\t"<<final[i][j]<<"\t";
        if(i==via[i][j])
            cout<<"-";
    }
}

```

```

        else
            cout<<ch[via[i][j]];
    }
}
cout<<"\n\n-----\n\n";
}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{
    int cost[MAX][MAX],distance[MAX],pred[MAX];
    int visited[MAX],count,mindistance,nextnode,i,j;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            if(G[i][j]==0)
                cost[i][j]=INF;
            else
                cost[i][j]=G[i][j];
    for(i=0;i<n;i++)
    {
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
    }
    distance[startnode]=0;
    visited[startnode]=1;
    count=1;
    while(count<n-1)
    {
        mindistance=INF;
        for(i=0;i<n;i++)
            if(distance[i]<mindistance&&!visited[i])
            {
                mindistance=distance[i];
                nextnode=i;
            }
        visited[nextnode]=1;
        for(i=0;i<n;i++)
            if(!visited[i])
                if(mindistance+cost[nextnode][i]<distance[i])
                {
                    distance[i]=mindistance+cost[nextnode][i];
                    pred[i]=nextnode;
                }
        count++;
    }
    for(i=0;i<n;i++)
        if(i!=startnode)
        {
            if(distance[i]!=-1)
            {
                cout<<"\nDistance of the node "<<i<<" = "<<distance[i];
                cout<<"\nPath = "<<i;
                j=i;
                do
                {
                    j=pred[j];
                    cout<<"<- "<<j;
                }while(j!=startnode);
                cout<<"\n";
            }
        }
    }
}

```

```

        else
        {
            cout<<"\nDistance of the node "<<i<<" = INF";
            cout<<"\nNo path";
        }
    }
    cout<<"\n-----\n\n";
}
void LinkState()
{
    int G[MAX][MAX],i,j,n,u;
    cout<<"Enter the number of vertices: ";
    cin>>n;
    cout<<"\nEnter the adjacency matrix:\n";
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            cin>>G[i][j];
    while(true)
    {
        cout<<"\nEnter the source/start node: ";
        cin>>u;
        dijkstra(G,n,u);
    }
}

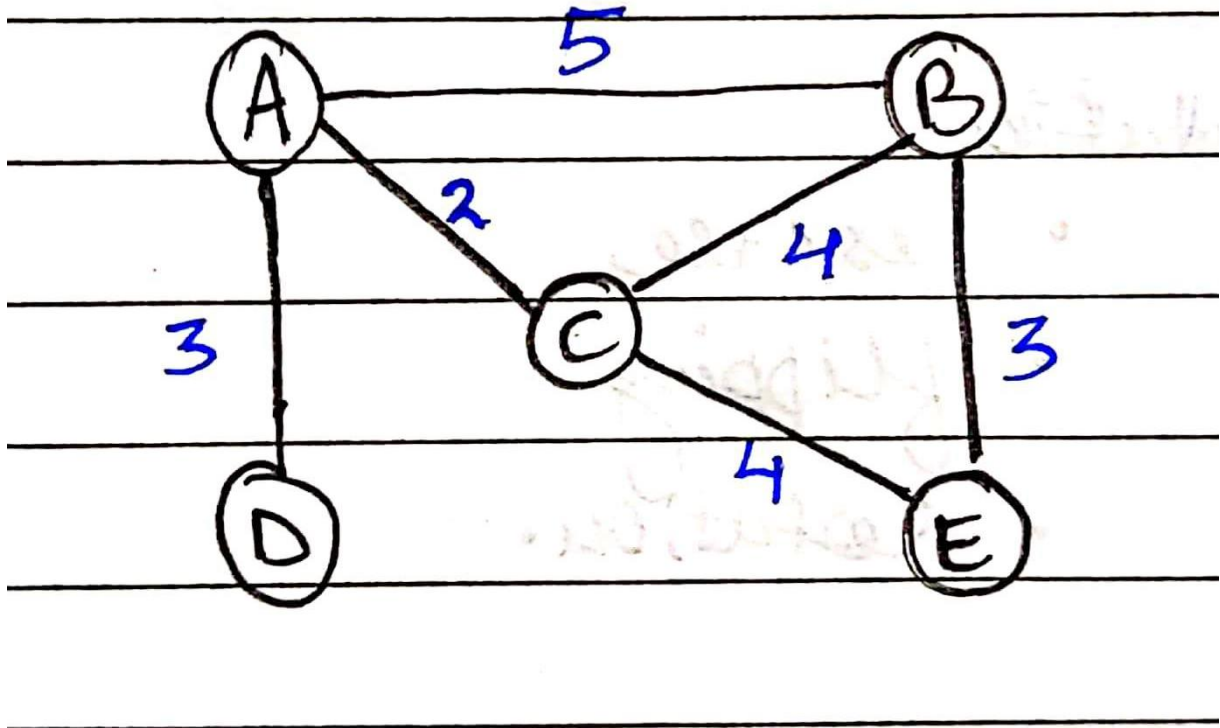
int main()
{
    system("cls");
    while(true)
    {
        cout<<"-----Unicast Routing Main Menu-----
\n        by [VIBHU KUMAR SINGH]\n\n";
        cout<<"1.Distance Vector Routing\n2.Link State Routing\n0. Exit\nEnter your choice
: ";
        int choice;
        cin>>choice;
        cout<<"-----
-\n\n";
        switch(choice)
        {
            case 1:
                DistanceVector();
                break;
            case 2:
                LinkState();
                break;
            case 0:
                exit(0);
                break;
            default:
                cout<<"\nInvalid Choice\n";
                break;
        }
    }
}

```

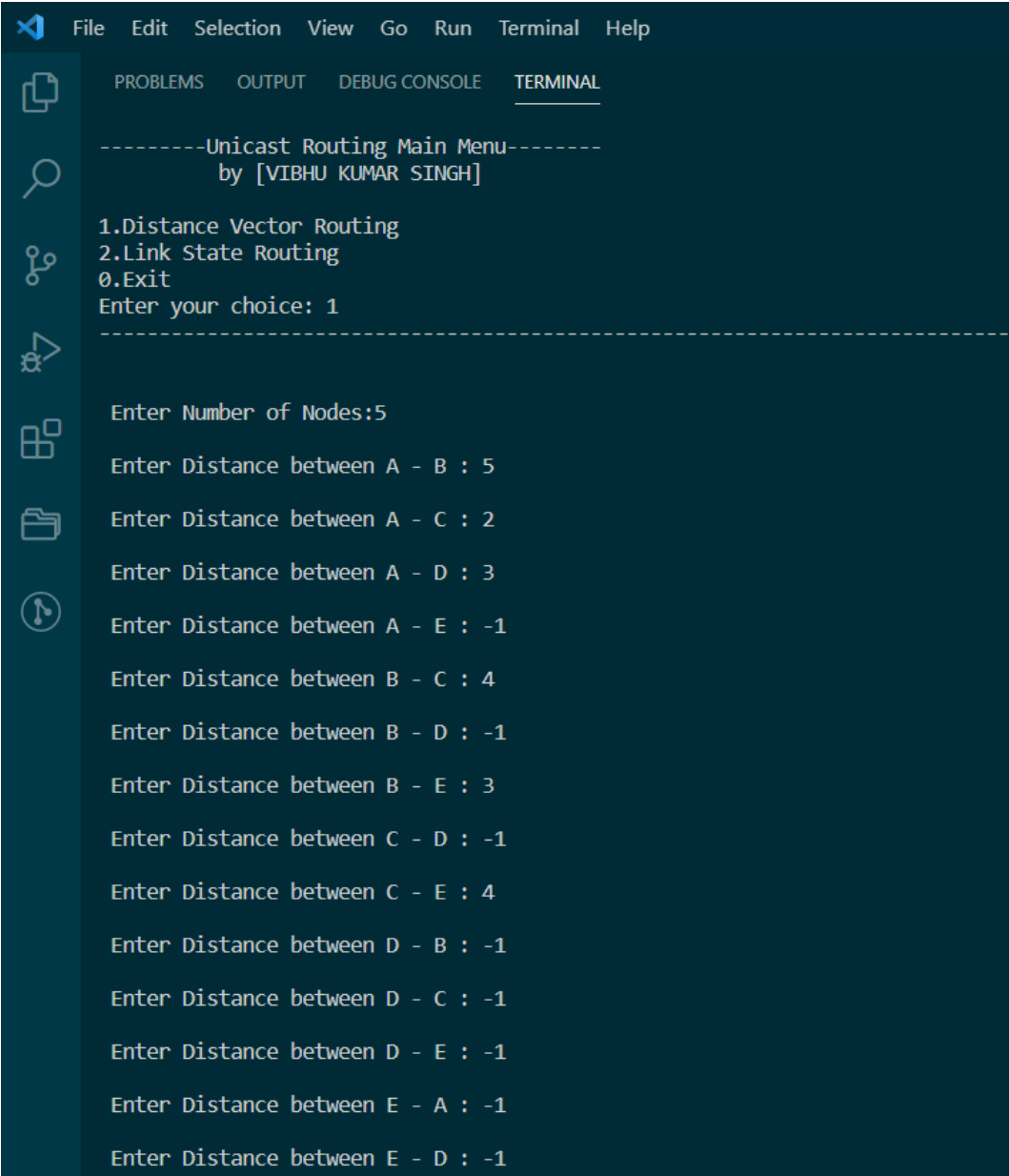
OUTPUT SCREENSHOTS:

a. Distance Vector Routing:

Input Graph:



Inputting:



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

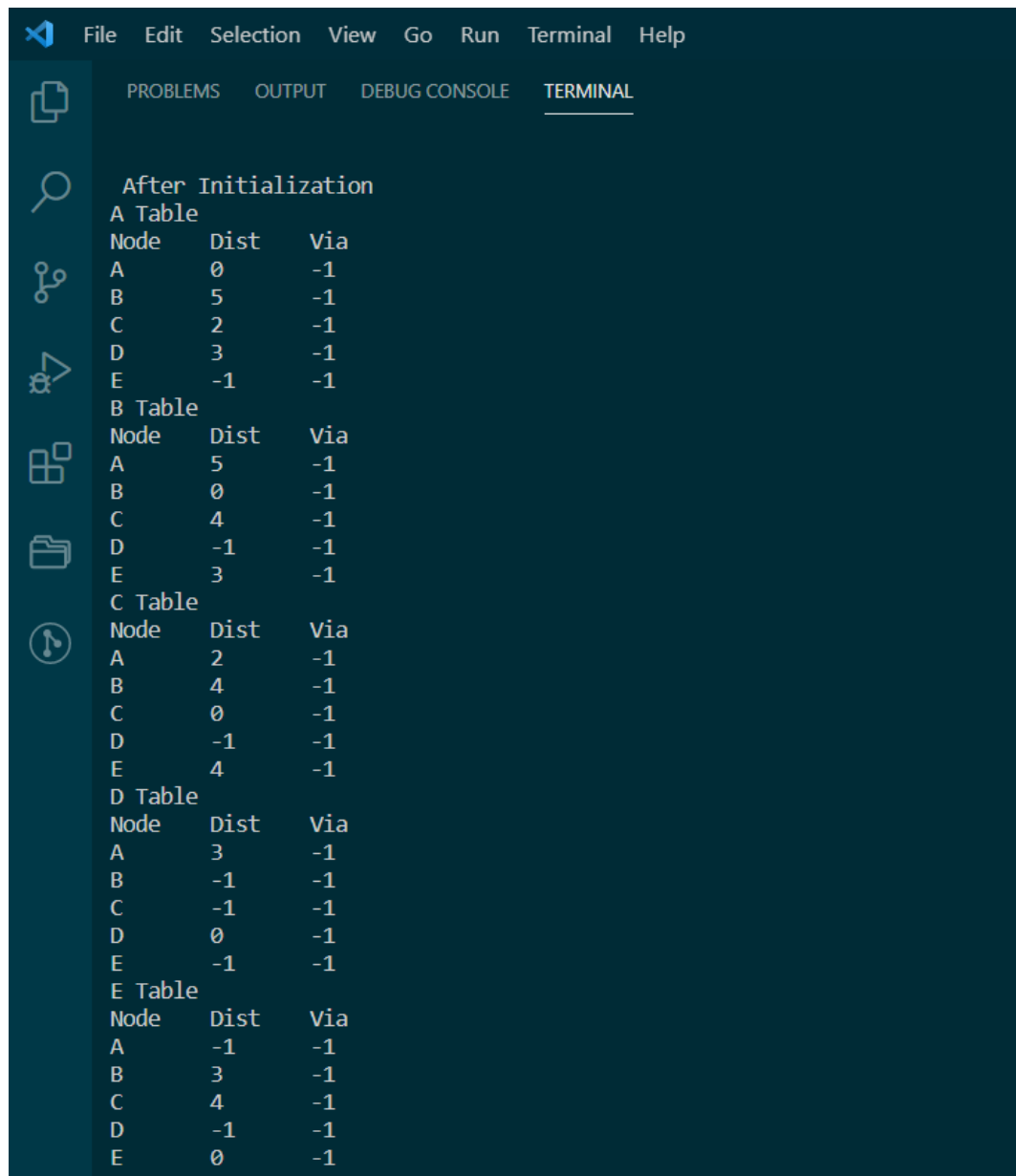
-----Unicast Routing Main Menu-----
by [VIBHU KUMAR SINGH]

1.Distance Vector Routing
2.Link State Routing
0.Exit
Enter your choice: 1
-----

Enter Number of Nodes:5

Enter Distance between A - B : 5
Enter Distance between A - C : 2
Enter Distance between A - D : 3
Enter Distance between A - E : -1
Enter Distance between B - C : 4
Enter Distance between B - D : -1
Enter Distance between B - E : 3
Enter Distance between C - D : -1
Enter Distance between C - E : 4
Enter Distance between D - B : -1
Enter Distance between D - C : -1
Enter Distance between D - E : -1
Enter Distance between E - A : -1
Enter Distance between E - D : -1
```

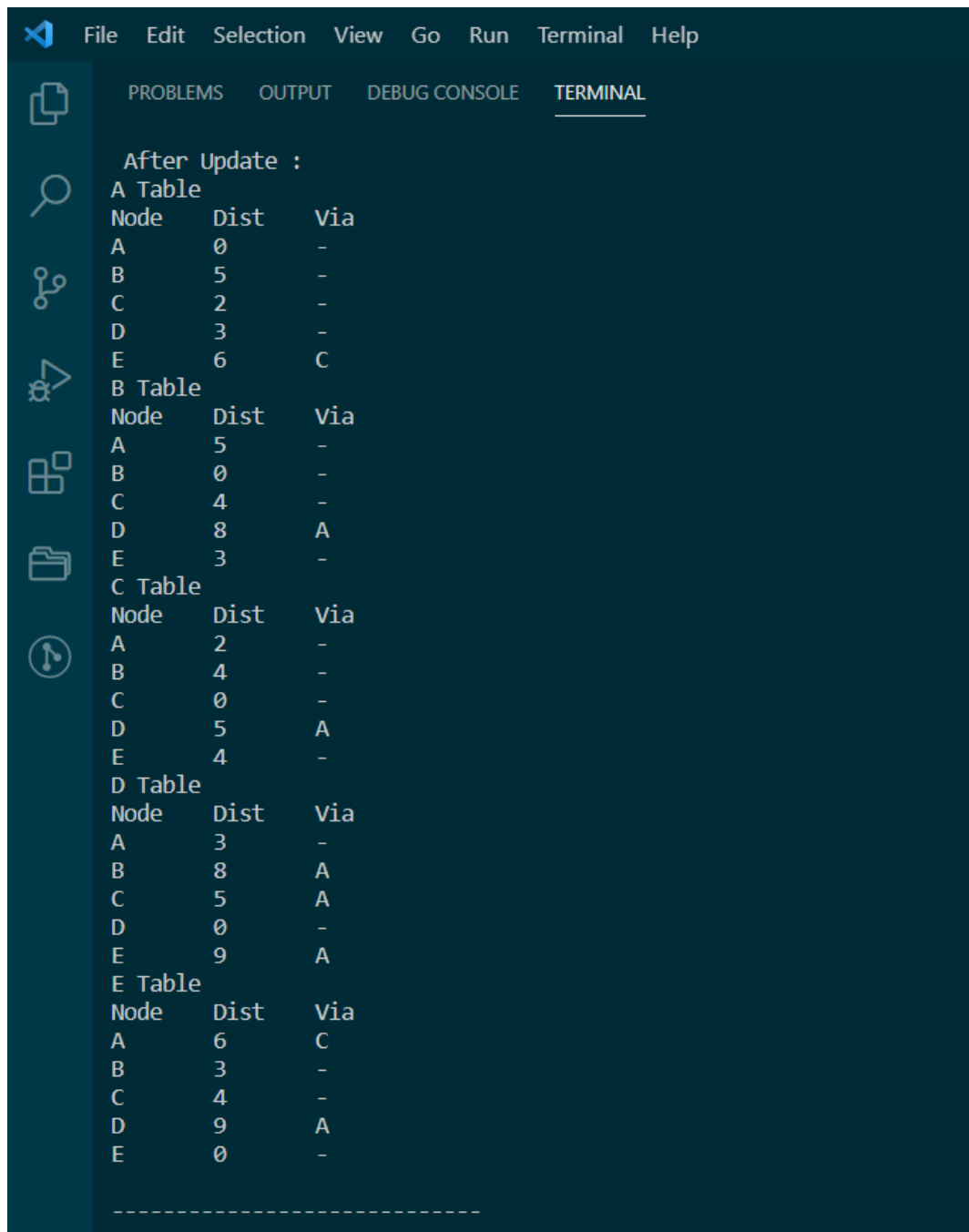
After Initialization:



The screenshot shows a VS Code terminal window with the 'TERMINAL' tab selected. The terminal output displays five tables (A, B, C, D, and E) showing network node information. Each table has three columns: Node, Dist, and Via. The data is as follows:

```
After Initialization
A Table
Node  Dist  Via
A      0    -1
B      5    -1
C      2    -1
D      3    -1
E     -1    -1
B Table
Node  Dist  Via
A      5    -1
B      0    -1
C      4    -1
D     -1    -1
E      3    -1
C Table
Node  Dist  Via
A      2    -1
B      4    -1
C      0    -1
D     -1    -1
E      4    -1
D Table
Node  Dist  Via
A      3    -1
B     -1    -1
C     -1    -1
D      0    -1
E     -1    -1
E Table
Node  Dist  Via
A     -1    -1
B      3    -1
C      4    -1
D     -1    -1
E      0    -1
```


After Updating:



The screenshot shows a VS Code terminal window with the following content:

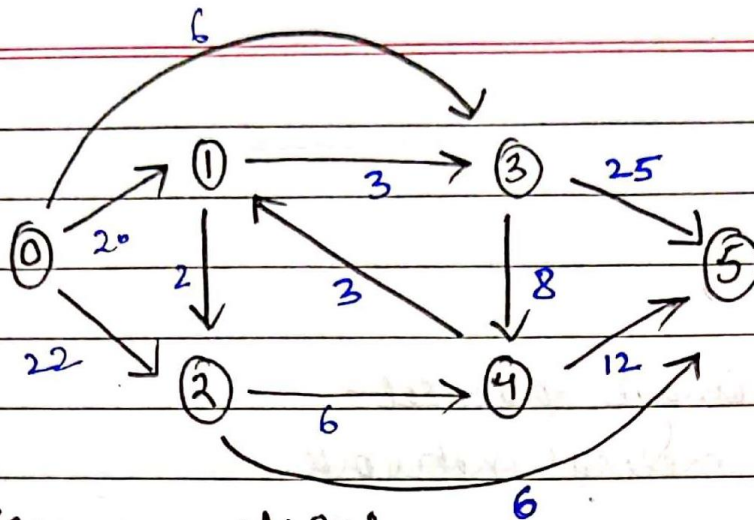
```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

After Update :
A Table
Node Dist Via
A 0 -
B 5 -
C 2 -
D 3 -
E 6 C
B Table
Node Dist Via
A 5 -
B 0 -
C 4 -
D 8 A
E 3 -
C Table
Node Dist Via
A 2 -
B 4 -
C 0 -
D 5 A
E 4 -
D Table
Node Dist Via
A 3 -
B 8 A
C 5 A
D 0 -
E 9 A
E Table
Node Dist Via
A 6 C
B 3 -
C 4 -
D 9 A
E 0 -

-----
```

b. Link State Routing:

Input Graph:



adjacency matrix:

	0	1	2	3	4	5	(-1 → ∞)
0	0	20	22	6	-1	-1	
1	-1	0	2	3	-1	-1	
2	-1	-1	0	-1	6	6	
3	-1	-1	-1	0	8	25	
4	-1	3	-1	-1	0	12	
5	-1	-1	-1	-1	-1	0	

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-----Unicast Routing Main Menu-----
by [VIBHU KUMAR SINGH]

1.Distance Vector Routing
2.Link State Routing
0.Exit
Enter your choice: 2
-----

Enter the number of vertices: 6

Enter the adjacency matrix:
0 20 22 6 9999 9999
9999 0 2 3 9999 9999
9999 9999 0 9999 6 6
9999 9999 9999 0 8 25
9999 3 9999 9999 0 12
9999 9999 9999 9999 9999 0

Enter the source/start node: 0

Distance of the node 1 = 17
Path = 1<-4<-3<-0

Distance of the node 2 = 19
Path = 2<-1<-4<-3<-0

Distance of the node 3 = 6
Path = 3<-0

Distance of the node 4 = 14
Path = 4<-3<-0

Distance of the node 5 = 25
Path = 5<-2<-1<-4<-3<-0
-----

File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Enter the source/start node: 1

Distance of the node 0 = 9999
Path = 0<-1

Distance of the node 2 = 2
Path = 2<-1

Distance of the node 3 = 3
Path = 3<-1

Distance of the node 4 = 8
Path = 4<-2<-1

Distance of the node 5 = 8
Path = 5<-2<-1
-----

Enter the source/start node: 2

Distance of the node 0 = 9999
Path = 0<-2

Distance of the node 1 = 9
Path = 1<-4<-2

Distance of the node 3 = 12
Path = 3<-1<-4<-2

Distance of the node 4 = 6
Path = 4<-2

Distance of the node 5 = 6
Path = 5<-2
-----
```

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

-----

Enter the source/start node: 3

Distance of the node 0 = 9999
Path = 0<-3

Distance of the node 1 = 11
Path = 1<-4<-3

Distance of the node 2 = 13
Path = 2<-1<-4<-3

Distance of the node 4 = 8
Path = 4<-3

Distance of the node 5 = 19
Path = 5<-2<-1<-4<-3

-----

Enter the source/start node: 4

Distance of the node 0 = 9999
Path = 0<-4

Distance of the node 1 = 3
Path = 1<-4

Distance of the node 2 = 5
Path = 2<-1<-4

Distance of the node 3 = 6
Path = 3<-1<-4

Distance of the node 5 = 11
Path = 5<-2<-1<-4

-----

Enter the source/start node: 5
PS C:\Users\Vibhu\Desktop\Winter Semester 20-21\NC\LAB\LAB 3> Input Graph
```