**Program Code:**

**User Input:**

#include <iostream>

#include <conio.h>

using namespace std;

class node

{

public:

node\* head=NULL; //head

node\* next;

string mssg;

void Enqueue(string msg)

{

node\* temp=new node;

temp->mssg=msg;

temp->next=NULL;

if(head==NULL)

{

head=temp;

}

else

{

node\* ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

string dequeue()

{

string MSG;

node \*temp=head;

MSG=head->mssg;

if(head->next==NULL)

return MSG;

else

{

head=head->next;

}

}

bool isEmpty()

{

if(head==NULL)

return true;

return false;

}

void show()

{

node \*ptr=head;

cout << endl;

while(ptr->next!=NULL)

{

cout << ptr->mssg << "\n";

ptr=ptr->next;

}

cout << ptr->mssg << "\n";

}

}N;

class Network

{

public:

string data;

// Pointers

Network \*modem;

Network \*up;

Network \*down;

Network \*left;

Network \*right;

Network()

{

modem=up=down=left=right= NULL;

data="\0";

}

void Creat\_Network()

{

string arr[4] = {"up","left","right","down"};

for (int i = 0; i < 5; ++i)

{

Network \*temp = new Network;

if (modem==NULL)

modem=temp;

else

{

if (arr[i-1]=="up")

{

modem->up = temp;

temp->data = "A";

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

{

Network \*temp1 = new Network;

if (arr[j] == "down")

{

temp->down = temp1;

temp1->data = "1.1.1.1";

}

else if (arr[j] == "up")

{

temp->up = temp1;

temp1->data = "2.2.2.2";

}

else if (arr[j] == "left")

{

temp->left= temp1;

temp1->data = "3.3.3.3";

}

else if (arr[j] == "right")

{

temp->right = temp1;

temp1->data = "4.4.4.4";

}

}

}

if (arr[i-1]=="down")

{

modem->down = temp;

temp->data = "D";

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

{

Network \*temp1 = new Network;

if (arr[j] == "up")

{

temp->up = temp1;

temp1->data = "225.100.0.1";

}

if (arr[j] == "left")

{

temp->left = temp1;

temp1->data = "230.0.0.1";

}

if (arr[j] == "right")

{

temp->right = temp1;

temp1->data = "235.35.35.35";

}

if (arr[j] == "down")

{

temp->down = temp1;

temp1->data = "233.33.0.3";

}

}

}

if (arr[i-1]=="left")

{

modem->left = temp;

temp->data = "B";

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

{

Network \*temp1 = new Network;

if (arr[j] == "up")

{

temp->up = temp1;

temp1->data = "154.68.1.1";

}

if (arr[j] == "left")

{

temp->left = temp1;

temp1->data = "169.0.0.1";

}

if (arr[j] == "right")

{

temp->right = temp1;

temp1->data = "129.1.1.1";

}

if (arr[j] == "down")

{

temp->down = temp1;

temp1->data = "191.68.1.1";

}

}

}

if (arr[i-1]=="right")

{

modem->right = temp;

temp->data = "C";

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

{

Network \*temp1 = new Network;

if (arr[j] == "up")

{

temp->up = temp1;

temp1->data = "192.168.1.1";

}

if (arr[j] == "left")

{

temp->left = temp1;

temp1->data = "195.168.0.10";

}

if (arr[j] == "right")

{

temp->right = temp1;

temp1->data = "200.1.1.1";

}

if (arr[j] == "down")

{

temp->down = temp1;

temp1->data = "220.20.20.20";

}

}

}

}//Else End

}//For loop End

}// End of Creat\_Network

void traverse(string IP[], int size)

{

Network \*ptr = modem;

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

{

string str,dummy;

str=dummy="\0";

dummy = IP[i];

int x=0;

while(dummy[x]!='.')

{

str = str + dummy[x];

++x;

}

// Ranges

if (str>="1" && str <="126")

{

ptr = modem->up;

}

else if (str>="127" && str <="191")

{

ptr = modem->left;

}

else if (str>="192" && str <="223")

{

ptr = modem->right;

}

else if (str>="224" && str <="239")

{

ptr = modem->down;

}

bool flag= false;

if (ptr->up->data == IP[i] || ptr->down->data == IP[i] || ptr->right->data == IP[i] || ptr->left->data == IP[i])

{

flag = true;

cout << "\nComputer Has Been Found SuccessFully!";

cout << "\nThe Message = \' " << N.dequeue() << " \' Has Been Delivered!" << endl;

}

else if (flag == false)

{

cout << "\n Computer Not Found!" << endl;

N.dequeue();

}

}

}

};

void InsertionSort (string Msg[], int Prt[], string IP[],int n)

{

int prt,j;

string msg;

string ip;

//insertion sort

for(int i=1;i<n;i++)

{

msg = Msg[i];

prt = Prt[i];

ip = IP[i];

j = i-1;

while(j>=0 && Prt[j]<prt)

{

Msg[j+1] = Msg[j];

Prt[j+1] = Prt[j];

IP[j+1] = IP[j];

j--;

}

Msg[j+1] = msg;

Prt[j+1] = prt;

IP[j+1] = ip;

}

cout << "\n\nSorted Based on Priority";

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

{

cout << "\n" << i+1 << ". " << Prt[i] << "\t" << Msg[i] << "\t" << IP[i];

N.Enqueue(Msg[i]);

}

cout << "\n" << endl;

}

int main()

{

cout<<"\n\n\t\t\tTHE NETWORK EMULATOR\nInput From User\n";

Network n;

n.Creat\_Network();

string s[10];

int p[10];

string ip[10];

int opt;

for(int i=0;i<10;i++)

{

s[i] = "\0";

p[i] = 0;

}

int i=0;

int j=0;

do

{

cout << "\n\nEnter a priority (-1 to exit) : ";

cin >> p[i];

if(p[i]!=-1)

{

cout << "Enter a message : ";

cin >> s[i];

cout << "Enter the IP : ";

cin >> ip[i];

i++;

j++;

}

else break;

}

while(p[i]!=-1);

cout << "\n\n-> Number of messages = " << j;

string message[j];

int priority[j];

string IP[j];

for(int k=0;k<j;k++)

{

message[k]=s[k];

priority[k]=p[k];

IP[k]=ip[k];

cout << "\n" <<k+1 << ". " << priority[k] << "\t" << message[k] << "\t" << IP[k];

}

InsertionSort(message,priority,IP,j);

cout << "\n The Queue = ";

N.show();

cout << "\n Looking For Computer...";

cout << "\n";

n.traverse(IP, j);

}

**File Handling:**

#include <iostream>

#include <conio.h>

#include <fstream>

using namespace std;

class node

{

public:

node\* head=NULL; //head

node\* next;

string mssg;

void Enqueue(string msg)

{

node\* temp=new node;

temp->mssg=msg;

temp->next=NULL;

if(head==NULL)

{

head=temp;

}

else

{

node\* ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

string dequeue()

{

string MSG;

node \*temp=head;

MSG=head->mssg;

if(head->next==NULL)

return MSG;

else

{

head=head->next;

}

}

bool isEmpty()

{

if(head==NULL)

return true;

return false;

}

void show()

{

node \*ptr=head;

cout << endl;

while(ptr->next!=NULL)

{

cout << ptr->mssg << " -> ";

ptr=ptr->next;

}

cout << ptr->mssg << "\n";

}

}N;

class Network

{

public:

string data;

// Pointers

Network \*modem;

Network \*up;

Network \*down;

Network \*left;

Network \*right;

Network()

{

modem=up=down=left=right= NULL;

data="\0";

}

void Creat\_Network()

{

string arr[4] = {"up","left","right","down"};

for (int i = 0; i < 5; ++i)

{

Network \*temp = new Network;

if (modem==NULL)

modem=temp;

else

{

if (arr[i-1]=="up")

{

modem->up = temp;

temp->data = "A";

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

{

Network \*temp1 = new Network;

if (arr[j] == "down")

{

temp->down = temp1;

temp1->data = "1.1.1.1";

}

else if (arr[j] == "up")

{

temp->up = temp1;

temp1->data = "2.2.2.2";

}

else if (arr[j] == "left")

{

temp->left= temp1;

temp1->data = "3.3.3.3";

}

else if (arr[j] == "right")

{

temp->right = temp1;

temp1->data = "4.4.4.4";

}

}

}

if (arr[i-1]=="down")

{

modem->down = temp;

temp->data = "D";

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

{

Network \*temp1 = new Network;

if (arr[j] == "up")

{

temp->up = temp1;

temp1->data = "225.100.0.1";

}

if (arr[j] == "left")

{

temp->left = temp1;

temp1->data = "230.0.0.1";

}

if (arr[j] == "right")

{

temp->right = temp1;

temp1->data = "235.35.35.35";

}

if (arr[j] == "down")

{

temp->down = temp1;

temp1->data = "233.33.0.3";

}

}

}

if (arr[i-1]=="left")

{

modem->left = temp;

temp->data = "B";

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

{

Network \*temp1 = new Network;

if (arr[j] == "up")

{

temp->up = temp1;

temp1->data = "154.68.1.1";

}

if (arr[j] == "left")

{

temp->left = temp1;

temp1->data = "169.0.0.1";

}

if (arr[j] == "right")

{

temp->right = temp1;

temp1->data = "129.1.1.1";

}

if (arr[j] == "down")

{

temp->down = temp1;

temp1->data = "191.68.1.1";

}

}

}

if (arr[i-1]=="right")

{

modem->right = temp;

temp->data = "C";

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

{

Network \*temp1 = new Network;

if (arr[j] == "up")

{

temp->up = temp1;

temp1->data = "192.168.1.1";

}

if (arr[j] == "left")

{

temp->left = temp1;

temp1->data = "195.168.0.10";

}

if (arr[j] == "right")

{

temp->right = temp1;

temp1->data = "200.1.1.1";

}

if (arr[j] == "down")

{

temp->down = temp1;

temp1->data = "220.20.20.20";

}

}

}

}//Else End

}//For loop End

}// End of Creat\_Network

void traverse(string IP[], int size)

{

Network \*ptr = modem;

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

{

string str,dummy;

str=dummy="\0";

dummy = IP[i];

int x=0;

while(dummy[x]!='.')

{

str = str + dummy[x];

++x;

}

// Ranges

if (str>="1" && str <="126")

{

ptr = modem->up;

}

else if (str>="127" && str <="191")

{

ptr = modem->left;

}

else if (str>="192" && str <="223")

{

ptr = modem->right;

}

else if (str>="224" && str <="239")

{

ptr = modem->down;

}

bool flag= false;

if (ptr->up->data == IP[i] || ptr->down->data == IP[i] || ptr->right->data == IP[i] || ptr->left->data == IP[i])

{

flag = true;

cout << "\nComputer Has Been Found SuccessFully!";

cout << "\nThe Message = \' " << N.dequeue() << " \' Has Been Delivered!" << endl;

}

else if (flag == false)

{

cout << "\n Computer Not Found!" << endl;

N.dequeue();

}

}

}

};

void InsertionSort (string Msg[], int Prt[], string IP[],int n)

{

int prt,j;

string msg;

string ip;

//insertion sort

for(int i=1;i<n;i++)

{

msg = Msg[i];

prt = Prt[i];

ip = IP[i];

j = i-1;

while(j>=0 && Prt[j]<prt)

{

Msg[j+1] = Msg[j];

Prt[j+1] = Prt[j];

IP[j+1] = IP[j];

j--;

}

Msg[j+1] = msg;

Prt[j+1] = prt;

IP[j+1] = ip;

}

cout << "\n\nSorted Based on Priority";

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

{

cout << "\n" << i+1 << ". " << Prt[i] << "\t" << Msg[i] << "\t" << IP[i];

N.Enqueue(Msg[i]);

}

cout << "\n" << endl;

}

int main()

{

cout<<"\n\n\t\t\tTHE NETWORK EMULATOR\nReading From a File\n";

Network n;

n.Creat\_Network();

string s[10];

int p[10];

string ip[10];

int opt;

for(int i=0;i<10;i++)

{

s[i] = "\0";

p[i] = 0;

ip[i]="\0";

}

int i=0;

int j=-1;

ifstream file("Net.txt");

while(!file.eof())

{

string a;

char b;

string xx;

string c;

getline(file,a); //mssg

int pr;

file>>b;

pr=int(b)-48; //priority

getline(file,xx);

getline(file,c); //IP

s[i]=a;

p[i]=pr;

ip[i]=c;

i++;

j++;

}

cout << "\n\n-> Number of messages = " << j;

string message[j];

int priority[j];

string IP[j];

for(int k=0;k<j;k++)

{

message[k]=s[k];

priority[k]=p[k];

IP[k]=ip[k];

cout << "\n" <<k+1 << ". " << priority[k] << "\t" << message[k] << "\t" << IP[k];

}

InsertionSort(message,priority,IP,j);

cout << "\n The Queue = ";

N.show();

cout << "\n Looking For Computer...";

cout << "\n";

n.traverse(IP, j);

}