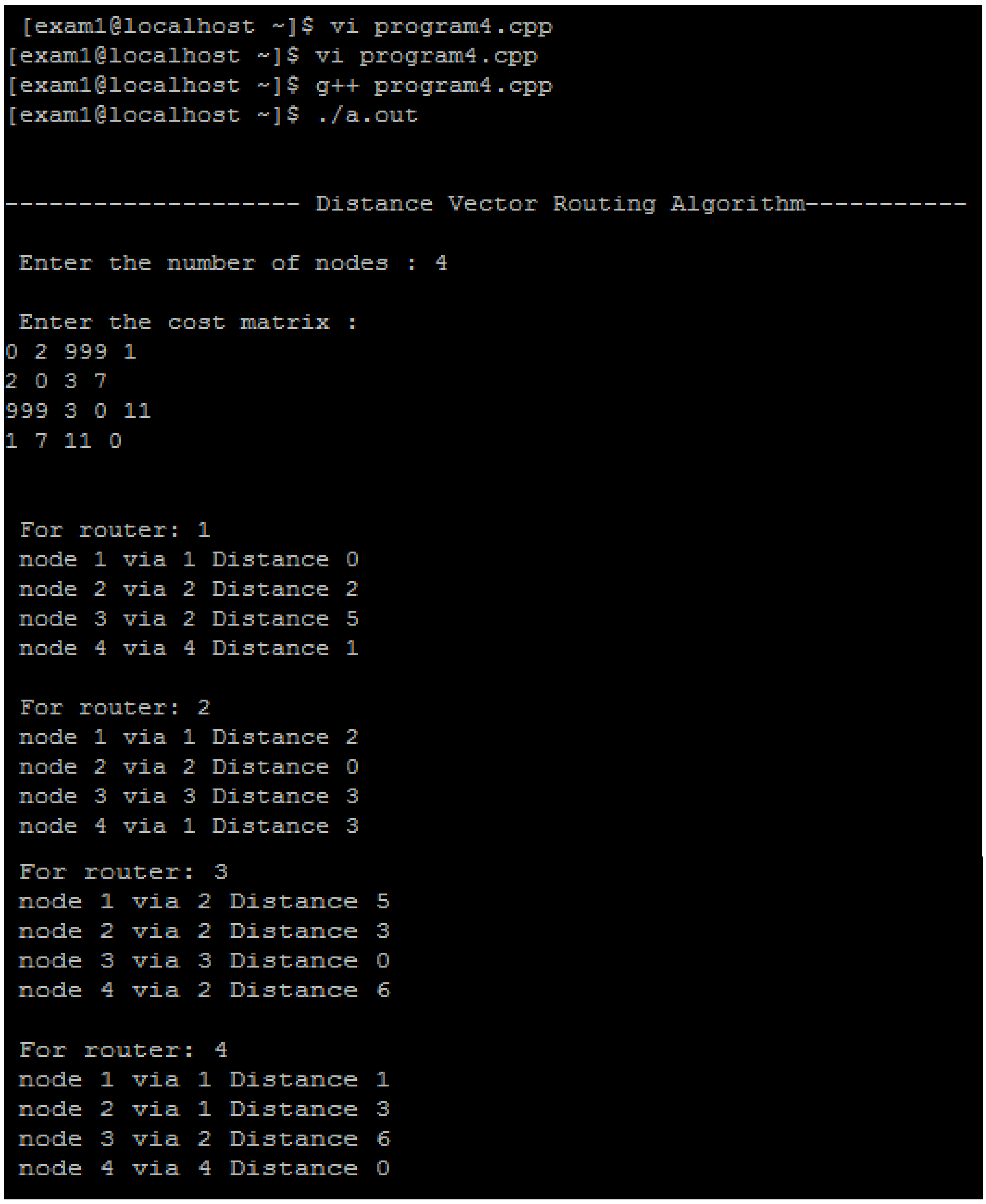
- It discovers that a link to a neighbor has gone down.

```
#include<stdio.h>
#include<iostream>
using namespace std;
struct node
{
   unsigned dist[6];
   unsigned from[6];
}DVR[10];
int main()
{
   cout<<"\n\n------------------- Distance Vector Routing Algorithm----------- ";
   int costmat[6][6];
   int nodes, i, j, k;
   cout<<"\n\n Enter the number of nodes : ";
   cin>>nodes; //Enter the nodes
   cout<<"\n Enter the cost matrix : \n" ;
   for(i = 0; i < nodes; i++)
    {
      for(j = 0; j < nodes; j++)
      {
         cin>>costmat[i][j];
         costmat[i][i] = 0;
         DVR[i].dist[j] = costmat[i][j]; //initialise the distance equal to cost matrix
         DVR[i].from[j] = j;
      }
   }
         for(i = 0; i < nodes; i++) //We choose arbitary vertex k and we calculate the
         //direct distance from the node i to k using the cost matrix and add the distance from k to
node j
         for(j = i+1; j < nodes; j++)
         for(k = 0; k < nodes; k++)
            if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])
            {   //We calculate the minimum distance
               DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];
               DVR[j].dist[i] = DVR[i].dist[j];
               DVR[i].from[j] = k;
               DVR[j].from[i] = k;
            }
      for(i = 0; i < nodes; i++)
      {
         cout<<"\n\n For router: "<<i+1;
         for(j = 0; j < nodes; j++)
            cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<" Distance "<<DVR[i].dist[j];
      }
   cout<<" \n\n ";
   return 0;
```

}

Output:



```
 [exam1@localhost ~]$ vi program4.cpp
[exam1@localhost ~]$ vi program4.cpp
[exam1@localhost ~]$ g++ program4.cpp
[exam1@localhost ~]$ ./a.out


-------------------- Distance Vector Routing Algorithm-----------

 Enter the number of nodes : 4

 Enter the cost matrix :
0 2 999 1
2 0 3 7
999 3 0 11
1 7 11 0


 For router: 1
 node 1 via 1 Distance 0
 node 2 via 2 Distance 2
 node 3 via 2 Distance 5
 node 4 via 4 Distance 1

 For router: 2
 node 1 via 1 Distance 2
 node 2 via 2 Distance 0
 node 3 via 3 Distance 3
 node 4 via 1 Distance 3
 For router: 3
 node 1 via 2 Distance 5
 node 2 via 2 Distance 3
 node 3 via 3 Distance 0
 node 4 via 2 Distance 6

 For router: 4
 node 1 via 1 Distance 1
 node 2 via 1 Distance 3
 node 3 via 2 Distance 6
 node 4 via 4 Distance 0
```

Least cost tree using link state protocol

5. **Creating a C++ program for the least cost tree using the Link State Routing algorithm requires simulating a network of nodes and performing the Link State algorithm. Below is a simplified example of how you can implement it:**

```cpp
#include <iostream>
#include <vector>
#include <climits>

using namespace std;

const int INF = INT_MAX;

class Network {
public:
    int numNodes;
    vector<vector<int>> costMatrix;

    Network(int nodes) : numNodes(nodes) {
        costMatrix.resize(nodes, vector<int>(nodes, INF));
    }

    void addLink(int node1, int node2, int cost) {
        costMatrix[node1][node2] = cost;
        costMatrix[node2][node1] = cost;
    }

    void printLeastCostTree(int source, const vector<int>& parent) {
        cout << "Least Cost Tree:" << endl;
        for (int i = 0; i < numNodes; ++i) {
            if (i != source) {
                cout << "Node " << i << " -> Node " << parent[i] << " (Cost: " << costMatrix[i][parent[i]] << ")" << endl;
            }
        }
    }

    void linkStateRouting(int source) {
        vector<int> distance(numNodes, INF);
        vector<bool> inTree(numNodes, false);
        vector<int> parent(numNodes, -1);
        distance[source] = 0;

        for (int i = 0; i < numNodes - 1; ++i) {
            int u = getMinDistanceVertex(distance, inTree);
            inTree[u] = true;

            for (int v = 0; v < numNodes; ++v) {
                if (!inTree[v] && costMatrix[u][v] != INF && distance[u] + costMatrix[u][v]
< distance[v]) {
                    parent[v] = u;
                    distance[v] = distance[u] + costMatrix[u][v];
                }
            }
        }

        printLeastCostTree(source, parent);
    }

    int getMinDistanceVertex(const vector<int>& distance, const vector<bool>& inTree)
{
        int minDistance = INF;
        int minVertex = -1;

        for (int v = 0; v < numNodes; ++v) {
            if (!inTree[v] && distance[v] < minDistance) {
                minDistance = distance[v];
                minVertex = v;
            }
        }

        return minVertex;
    }
};

int main() {
    int numNodes = 4;
    Network network(numNodes);

    // Add links with their costs
    network.addLink(0, 1, 4);
    network.addLink(0, 2, 2);
    network.addLink(1, 2, 5);
    network.addLink(1, 3, 10);
    network.addLink(2, 3, 1);
    int sourceNode = 0;
    network.linkStateRouting(sourceNode);

    return 0;
}
```

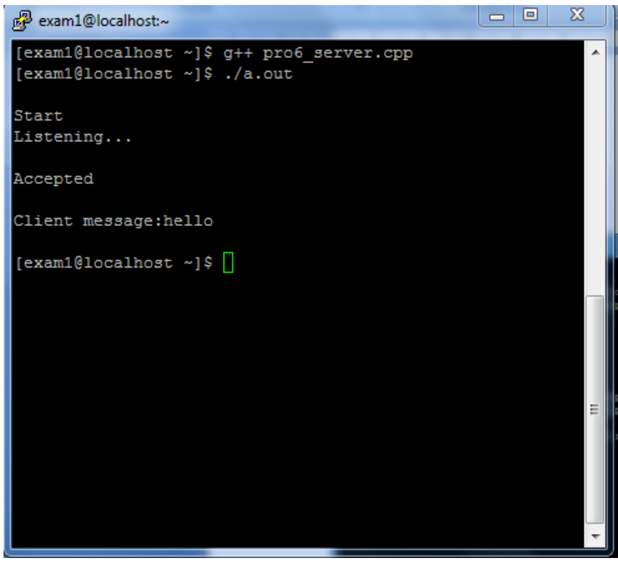
6. To write echo client-server application using TCP.

```
// TCP server side

//Include headers
#include<stdio.h>
#include<netinet/in.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<unistd.h>
//Define server port
#define SERV_TCP_PORT 5035
int main(int argc,char**argv)
{
  //variable declaration
      int sockfd,newsockfd;
      socklen_t clength;
      struct sockaddr_in serv_addr,cli_addr;
      char buffer[4096];


   // create socket
      sockfd=socket(AF_INET,SOCK_STREAM,0);

   //Initialize server addres structure
      serv_addr.sin_family=AF_INET;
      serv_addr.sin_addr.s_addr=INADDR_ANY;
      serv_addr.sin_port=htons(SERV_TCP_PORT);

      printf("\nStart");
 // bind socket
      bind(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
      printf("\nListening...");
      printf("\n");
//Listen incoming connection
      listen(sockfd,5);
```

```
// accept connection from client
        clength=sizeof(cli_addr);
        newsockfd=accept(sockfd,(struct sockaddr*)&cli_addr,&clength);
        printf("\nConnection Accepted");
        printf("\n");
//Read client message
        read(newsockfd,buffer,4096);
        printf("\nClient message:%s",buffer);
//echo back to client
        write(newsockfd,buffer,4096);
        printf("\n");
//close sockets
        close(sockfd);
        close(newsockfd);
        return 0;
}


//TCP client side
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<unistd.h>
#include<arpa/inet.h>
#define SERV_TCP_PORT 5035
int main(int argc,char*argv[])
{
        int sockfd;
        struct sockaddr_in serv_addr;
        struct hostent *server;
        char buffer[4096];

        sockfd=socket(AF_INET,SOCK_STREAM,0);
        serv_addr.sin_family=AF_INET;
        serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
        serv_addr.sin_port=htons(SERV_TCP_PORT);

        printf("\nReady for sending...");
        connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
        printf("\nEnter the message to send\n");
        printf("\nClient: ");
        fgets(buffer,4096,stdin);
        write(sockfd,buffer,4096);
        printf("Serverecho:%s",buffer);
        printf("\n");
        close(sockfd);
        return 0;
}
```

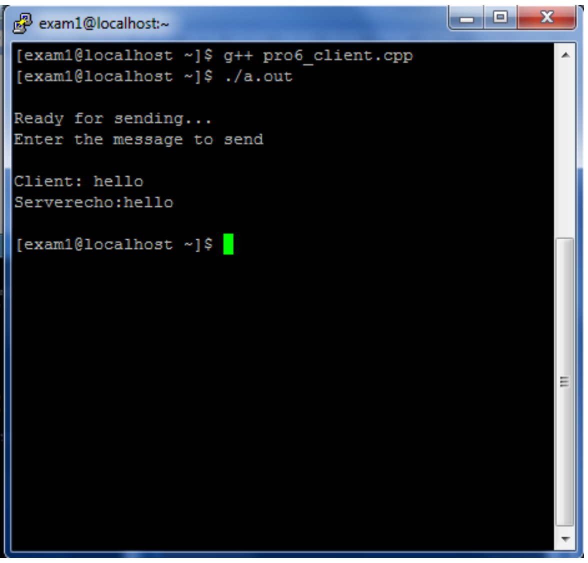
Output:

At server terminal



```
[exam1@localhost ~]$ g++ pro6_server.cpp
[exam1@localhost ~]$ ./a.out

Start
Listening...

Accepted

Client message:hello

[exam1@localhost ~]$
```

At client terminal



```
[exam1@localhost ~]$ g++ pro6_client.cpp
[exam1@localhost ~]$ ./a.out

Ready for sending...
Enter the message to send

Client: hello
Serverecho:hello

[exam1@localhost ~]$
```