Use Network Utility Command like ping, ipconfig, netstat, tracert to observe the network details.

1) The `ping` command is used to check the connectivity between your computer and another device on the network. This command will send ICMP Echo Request packets to `www.google.com` and wait for replies. It helps in determining if the destination is reachable and how long it takes to get a response.

ping www.google.com

2) The `ipconfig` (on Windows) or `ifconfig` (on Linux/Unix) command is used to display the network configuration details of your computer. This command will show IP addresses, subnet masks, default gateways, and other network settings for all network interfaces on your computer.

Ipconfig ifconfig

3) The `netstat` command is used to display network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.This command will show all active network connections and listening ports, along with the protocol (TCP/UDP), local and foreign addresses, and the connection state.

netstat -an

4) The `tracert` (on Windows) or `traceroute` (on Linux/Unix) command is used to trace the path that packets take from your computer to a destination host. This command will display each hop along the route to the destination, showing the time it takes for each hop. This can help in identifying where delays or packet loss are occurring along the path.

tracert www.google.com

traceroute www.google.com

To implement Hamming Code using C and C++.

```
#include<iostream>
using namespace std;
const int N = 32;
int arr[N];
int hash_binary[100][32];

int power(int a, int b)
{
    if(b == 0)
    {
        return 1;
    }
    int ans = 1;
    while(b--)
    {
        ans *= a;
    }
    return ans;
```

```
}

int count_no_of_1(int total_bits,int index,string* arr1)
{
    int count = 0;
    for(int i=1;i<=total_bits;i++)
    {
        if(hash_binary[i][index] == 1)
        {
            if(arr1[i] == "1")
            {
                count++;
            }
        }
    }
    // only for checking
    for(int i=total_bits;i>=1;i--)
    {
        cout<<arr1[i]<<" ";
    }
    cout<<endl;
    // only for checking
    return count;
}

int main()
{
    for(int i=1;i<=100;i++)
    {
        int temp_index1 = 1;
        int n = i;
        while (n > 0)
        {
            hash_binary[i][temp_index1++] = n % 2;
            n /= 2;
        }
    }
    string myinput;
    cin>>myinput;
    int countmyinput = 0;
    for(int i=0;i<myinput.length();i++)
    {
        countmyinput++;
    }
    int m = countmyinput;
    cout<<"m(data bits) = "<<m<<endl;
    int r;
    for(int i=0;i<100;i++)
    {
        if(power(2,i) >= (m+i+1))
        {
            r = i;
            break;
        }
```

```
    }
    cout<<"r(redundant bits) = "<<r<<endl;
    int total_bits = m+r;
    cout<<"Total number of bits = "<<total_bits<<endl;

    string arr1[total_bits+1];
    int temp1 = 0,temp2 = m-1;

    for(int i=1;i<=(total_bits);i++)
    {
        if(power(2,temp1) == i)
        {
            arr1[i] = "r"+ to_string(i);
            temp1++;
        }
        else
        {
            arr1[i] = myinput[temp2--];
        }
    }
    for(int i=1;i<=total_bits;i++)
    {
        cout<<arr1[i]<<" ";
    }
    cout<<endl;
    for(int i=total_bits;i>=1;i--)
    {
        cout<<arr1[i]<<" ";
    }
    cout<<endl;
    int count_of_1, count_of_r = 1;
    int error[r],error_temp = 0;
    for(int i=1;i<=total_bits;i++)
    {
        if(arr1[i] != "1" && arr1[i] != "0")
        {
            count_of_1 = count_no_of_1(total_bits,count_of_r,arr1);
            cout<<arr1[i]<<" = "<<count_of_1%2<<endl;
            string temp = to_string(count_of_1 % 2);
            arr1[i] = temp;
            error[error_temp++] = count_of_1 % 2;
            count_of_r++;
        }
    }
    for(int i=total_bits;i>=1;i--)
    {
        cout<<arr1[i]<<" ";
    }
    for(int i=0;i<r;i++)
    {
        if(error[i] == 1)
        {
            cout<<"\nerror occured \n";
            break;
```

```
        }
    }
    int sum = 0;
    for(int i=0;i<r;i++)
    {
        sum += error[i]*power(2,i);
    }
    cout<<sum<<" th position \n";
    for(int i=total_bits;i>=1;i--)
    {
        if(i == sum)
        {
            if(arr1[i] == "1")
            {
                arr1[i] = "0";
            }
            else
            {
                arr1[i] = "1";
            }
        }
        cout<<arr1[i]<<" ";
    }
    cout<<"\n Error Resolved";

    return 0;
}
```

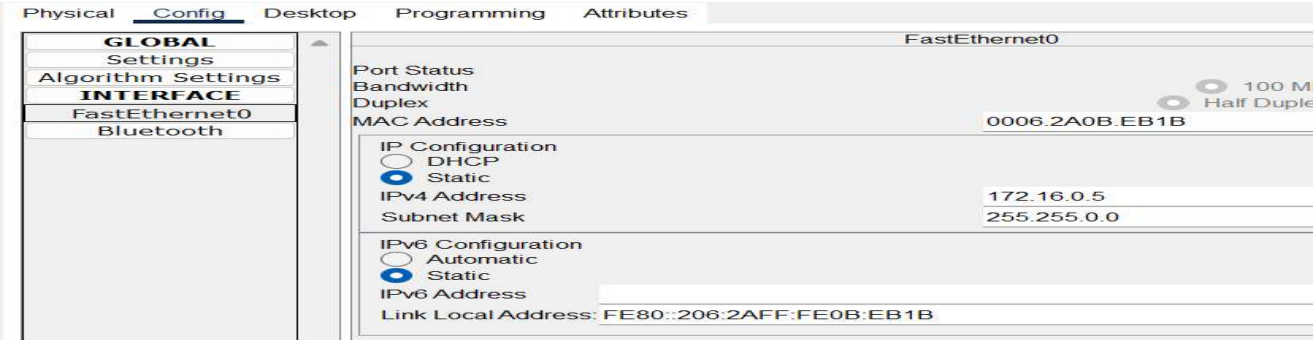
To implement Dijkstra's Routing algorithm using backtracking approach.

```
#include<stdio.h>
#include<stdlib.h>

#define INFINITE 999

//function prototype
void dijkstraBacktrack(int n,  int cost[10][10], int s, int t, int dist[10], int
currentDist, int *minDist);

int main() {
    int i, j, n, s, cost[10][10], dist[10];
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    printf("Enter the cost:\n");
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
            scanf("%d", &cost[i][j]);

            //if cost is zero then it is considered to br infinite
            if (cost[i][j] == 0) {
                cost[i][j] = INFINITE;
            }
        }
```

```
    }
    printf("Enter the source node: ");
    scanf("%d", &s);

    // Initialize minimum distance to a large value
    int minDist = INFINITE;

    // function call
    dijkstraBacktrack(n, cost, s, s, dist, 0, &minDist);

    // it print the shortest path from the source node
    printf("Shortest path from %d is:\n", s);
    for (i = 1; i <= n; i++) {
        if (s != i) {
            printf("%d->%d=%d\n", s, i, dist[i]);
        }
    }
    return 0;
}

void dijkstraBacktrack(int n,  int cost[10][10], int s, int t, int dist[10], int
currentDist, int *minDist)
{
    if (s == t) {
        if (currentDist < *minDist) {
            *minDist = currentDist;
            for (int i = 1; i <= n; i++) {
                dist[i] = cost[s][i];
            }
        }
        return;
    }

    for (int i = 1; i <= n; i++) {
        if (cost[s][i] != INFINITE) {
            int tempDist = cost[s][i];
            // Marking the current edge as visited
            cost[s][i] = INFINITE;

            // function call to explore the next node
            dijkstraBacktrack(n, cost, i, t, dist, currentDist + tempDist, minDist);

            // Backtrack: restore the cost matrix
            cost[s][i] = tempDist;
        }
    }
}
```

Use traffic monitoring tool Wireshark to observe network traffic with packet details.

Configure router. Configure network using Cisco Packet Tracer software and show packet transmission from source to destination



To implement Go_back_n sliding window protocol.

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

int main()
{
    cout << "Enter the window size: ";
    int window_size;
    cin >> window_size;
    for (int i = 0; i < window_size; i++)
    {
        cout << "Frame " << i << "has been transmitted successfully ." << endl;
    }

    int sent;
    cout << "Enter the number of acknowledgement recieved" << endl;
    cin >> sent;
    for (int i = sent; i < window_size; i++)
    {
        cout << "Frame " << i << "has been re transmitted successfully" << endl;
    }
    return 0;
}
```

Client server communication using socket programming

```
import java.io.*;
import java.net.*;
public class MyServer {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(6665);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream());
String  str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
}catch(Exception e){System.out.println(e);}
}
}
import java.io.*;
import java.net.*;
public class MyClient {
```

```
public static void main(String[] args) {
try{
Socket s=new Socket("localhost",6665);
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
dout.writeUTF("Hello Student");
dout.flush();
dout.close();
s.close();
}catch(Exception e){System.out.println(e);}
}
}
```

Simulating IoT environment using Cisco Packet Tracer.