**Exp13 Code:- (DNS Lookup)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.Net.Sockets;

namespace ConsoleApp13{

internal class Program {

static void Main(string[] args) {

IPHostEntry IPHost = Dns.GetHostEntry("www.hotmail.com");

Console.WriteLine(IPHost.HostName);

string[] aliases = IPHost.Aliases;

Console.WriteLine(aliases.Length);

IPAddress[] addr = IPHost.AddressList;

Console.WriteLine(addr.Length);

for (int i = 0; i < addr.Length; i++)

{

Console.WriteLine(addr[i]);

} } } }

Output:-



**Code:- (retrieves Basic information about networks)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.NetworkInformation;

namespace ConsoleApp13

{

internal class Program

{

static void Main(string[] args)

{

NetworkInterface[] niArr = NetworkInterface.GetAllNetworkInterfaces();

Console.WriteLine("Retriving basic information of network.\n\n");

foreach (NetworkInterface tempNetworkInterface in niArr)

{

Console.WriteLine("Network Discription : " + tempNetworkInterface.Description);

Console.WriteLine("Network ID : " + tempNetworkInterface.Id);

Console.WriteLine("Network Name : " + tempNetworkInterface.Name);

Console.WriteLine("Network interface type : " + tempNetworkInterface.NetworkInterfaceType.ToString());

Console.WriteLine("Network Operational Status : " + tempNetworkInterface.OperationalStatus.ToString());

Console.WriteLine("Network Spped : " + tempNetworkInterface.Speed);

Console.WriteLine("Support Multicast : " + tempNetworkInterface.SupportsMulticast);

Console.WriteLine();

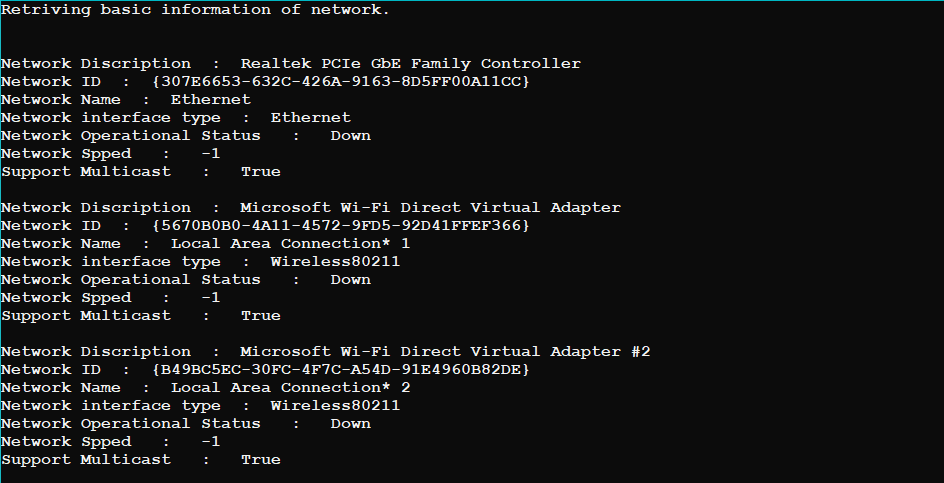
}

}

}

}

**Output:-**

  
**Code:- (monitors network availability and network address)**

using System;

using System.Net;

using System.Net.NetworkInformation;

using System.Threading;

namespace ConsoleApplication7{

class Program {

static void Main() {

Console.WriteLine("Initial Network Information:");

DisplayNetworkInformation();

while (true) {

Thread.Sleep(5000);

Console.WriteLine("\nChecking for Network Changes...");

if (NetworkInformationChanged()) {

Console.WriteLine("Network Changes Detected!");

DisplayNetworkInformation();

} } }

static void DisplayNetworkInformation() {

string hostName = Dns.GetHostName();

Console.WriteLine("Hostname: " + hostName);

IPHostEntry hostEntry = Dns.GetHostEntry(hostName);

foreach (IPAddress ipAddress in hostEntry.AddressList) {

Console.WriteLine("IP Address: " + ipAddress);

}

NetworkInterface[] networkInterfaces = NetworkInterface.GetAllNetworkInterfaces();

Console.WriteLine("Network Interfaces:");

foreach (NetworkInterface networkInterface in networkInterfaces) {

Console.WriteLine("Interface: " + networkInterface.Name);

Console.WriteLine("Status: " + networkInterface.OperationalStatus);

Console.WriteLine("Speed: " + networkInterface.Speed + " bytes/s");

} }

static bool NetworkInformationChanged() {

string currentHostName = Dns.GetHostName();

IPHostEntry currentHostEntry = Dns.GetHostEntry(currentHostName);

IPAddress[] currentAddresses = currentHostEntry.AddressList;

if (currentAddresses.Length != Dns.GetHostEntry("").AddressList.Length) {

return true;

}

NetworkInterface[] currentNetworkInterfaces = NetworkInterface.GetAllNetworkInterfaces();

NetworkInterface[] previousNetworkInterfaces = GetPreviousNetworkInterfaces();

if (currentNetworkInterfaces.Length != previousNetworkInterfaces.Length)

{

return true;

}

return false;

}

static NetworkInterface[] GetPreviousNetworkInterfaces()

{

NetworkInterface[] previousNetworkInterfaces = (NetworkInterface[])NetworkInterface.GetAllNetworkInterfaces().Clone();

return previousNetworkInterfaces;

}

}

}

Output:-

