```
// SL.No.: - 31
// Admission No.: - 21JE0269
// Name: - Chotaliya Zeel Vijaybhai

// #include <iostream>
// #include <string>
// #include <limits>
// #include <vector>

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

// Implementing Application Layer.
class Application {
        private:
        string appLayer_message;

        public:

        /*
                get function - To update appLayer_message variable with
                        message from user or trasport layer.
        */
        void get(string message, bool txrx){
                appLayer_message = message;
        }

        /*
                print function - To print updated message and return it.
        */
        string print(){
                cout << "Message at Application layer: " << appLayer_message << endl;
                return appLayer_message;
        }
```

```
};

// Implementing Transport Layer.
class Transport {
        private:
        string header = "TL";
        string transpLayer_message;

        public:

        /*
                get function - To append or remove header from message.
                If txrx == 1 => append header.
                If txrx == 0 => remove header.
        */
        void get(string message, bool txrx){
                if(txrx == 1){
                        transpLayer_message = header + message;
                }else{
                        transpLayer_message = message.substr(2);
                }
        }


        /*
                print function - To print updated message and return it.
        */
        string print(){
                cout << "Message at Transport layer: " << transpLayer_message << endl;
                return transpLayer_message;
        }
};

// Implementing Network Layer.
class Network {
        private:
```

```
		string header = "NL";
		string nwtLayer_message;

		public:

		/*
			get function - To append or remove header from message.
			If txrx == 1 => append header.
			If txrx == 0 => remove header.
		*/
		void get(string message, bool txrx){
			if(txrx == 1){
				nwtLayer_message = header + message;
			}else{
				nwtLayer_message = message.substr(2);
			}
		}


		/*
			print function - To print updated message and return it.
		*/
		string print(){
			cout << "Message at Network layer: " << nwtLayer_message << endl;
			return nwtLayer_message;
		}
};

// Implementing DataLink Layer.
class DataLink {
		private:
		string header = "DL";
		string dlnkLayer_message;

		public:
```

```
        /*
                get function - To append or remove header from message.
                If txrx == 1 => append header.
                If txrx == 0 => remove header.
        */
        void get(string message, bool txrx){
                if(txrx == 1){
                        dlnkLayer_message = header + message;
                }else{
                        dlnkLayer_message = message.substr(2);
                }
        }


        /*
                print function - To print updated message and return it.
        */
        string print(){
                cout << "Message at DataLink layer: " << dlnkLayer_message << endl;
                return dlnkLayer_message;
        }
};

// Implementing Physical Layer.
class Physical {
        private:
        vector<int> dataStream;
        string phyLayer_message;

        public:

        /*
                get function - To To convert message to dataStream or vice versa.
                If txrx == 1 => message to dataStream.
                If txrx == 0 => dataStream to message.
        */
```

```
void get(string message, vector<int> data, bool txrx){
		if(txrx == 1){
			dataStream.clear();
			for(char ch : message){
				dataStream.push_back((int)ch);
			}
		}else{
			phyLayer_message.clear();
			for(int ascii : data){
				phyLayer_message += (char)ascii;
			}
		}
}

/*
		print function - To print dataStream and message and return it.
		If txrx == 1 => printing dataStream.
		If txrx == 0 => printing data and phyLayer_message.
*/

// If txrx == 1, calling below function for printing.
vector<int> print_data(){
		cout << "Message at Physical layer: " << endl;
		cout << "ASCII: ";
		for(int ascii : dataStream){
			cout << ascii << ' ';
		}
		cout << endl;
		return dataStream;
}

// If txrx == 0, calling below function for printing.
string print_message(){
		cout << "Message at Physical layer: " << endl;
		cout << "ASCII: ";
		for(char ch : phyLayer_message){
```

```
                        cout << (int)ch << ' ';
                    }
                    cout << endl;
                    cout << "msg: " << phyLayer_message << endl;
                    return phyLayer_message;
                }
};

int main(){

        bool txrx;

        // If txrx is 1 - means message has to trasmit.
        // If txrx is 0 - means message has to receive.

        // Declaring message variable, dataStream vector and all class object varaible.
        string message;
        vector<int> dataStream;
        Application appLayer;
        Transport transpLayer;
        Network nwtLayer;
        DataLink dlnkLayer;
        Physical phyLayer;

        txrx = 1;
        cout << "--------Transmitting message--------" << endl;

                // Taking message from user.
                cout << "Enter Message: " << endl;
                getline(cin, message);
                //cout << message << endl;

                // Transmitting message to Application Layer.
                appLayer.get(message, txrx);
                message = appLayer.print();
```

```
                // Transmitting message to Transport Layer.
                transpLayer.get(message, txrx);
                message = transpLayer.print();

                // Transmitting message to Network Layer.
                nwtLayer.get(message, txrx);
                message = nwtLayer.print();

                // Transmitting message to DataLink Layer.
                dlnkLayer.get(message, txrx);
                message = dlnkLayer.print();

                // Tranmitting message to Physical Layer.
                dataStream.clear();
                phyLayer.get(message, dataStream, txrx);
                dataStream = phyLayer.print_data();

cout << endl;
txrx = 0;
cout << "--------Receiving message--------" << endl;

                // Receiving message at physical Layer.
                phyLayer.get(message, dataStream, txrx);
                message = phyLayer.print_message();

                // Receiving messaga at DataLink Layer.
                dlnkLayer.get(message, txrx);
                message = dlnkLayer.print();

                // Receiving messaga at Network Layer.
                nwtLayer.get(message, txrx);
                message = nwtLayer.print();

                // Receiving messaga at Transport Layer.
                transpLayer.get(message, txrx);
                message = transpLayer.print();
```

```
                    // Receiving messaga at Application Layer.
                    appLayer.get(message, txrx);
                    message = appLayer.print();

                    // Message received by user.

        return 0;
}
```