

## TABLE OF CONTENTS

Sno.	Name of the experiment	Date	Page no.	Sign
1.	Implement the data link layer framing methods such as character, character-stuffing and bit stuffing			
2.	Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CRC CCIP			
3.	Develop a simple data link layer that performs the flow control using the sliding window protocol, and Loss recovery using G0-Back-N mechanism			
4.	Implement Dijkstra's algorithm to compute the shortest path through a network			
5.	Implement distance vector routing algorithm for obtaining routing tables at each node.			
6.	Write a program for congestion control using Leaky bucket algorithm.			
7.	Write a program for IP Fragmentation			
8.	Wireshark i)packet capturing using Wireshark ii)Starting Wire Shark iii) Viewing Captured Traffic iv)Analysis and Statistics & Filters			

Sno.	Name of the experiment	Date	Page no.	Sign
9	Implement data encryption and data decryption			
10	Implement data encryption and data decryption			
11	Write a program for frame sorting technique used in buffers			

1) Implement the data link layer framing methods such as character, character-stuffing and bit stuffing

i) Bit stuffing:

```
#include <stdio.h>

int main()
{
    char arr[100];
    int n, i, j;
    printf("\nEnter number of bits :");
    scanf("%d", &n);
    printf("\nEnter bit string :");
    scanf("%s", arr);
    int count = 0;
    for (i = 0; i < n; i++)
    {
        if (arr[i] == '1')
        {
            count++;
            if (count == 5)
            {
                n = n + 1;
                for (j = n - 1; j > i; j--)
                {
```

```
        arr[j] = arr[j - 1];
    }
    arr[j + 1] = '0';
}
}
else if (arr[i] == '0')
{
    count = 0;
}
}
printf("\nAfter stuffing the string is : %s", arr);
for (i = 0; i < n; i++)
{
    if (arr[i] == '1')
    {
        count++;
        if (count == 5)
        {
            n = n - 1;
            for (j = i + 1; j < n; j++)
            {
                arr[j] = arr[j + 1];
            }
            arr[j] = '\0';
        }
    }
}
```

```
    else if (arr[i] == '0')
    {
        count = 0;
    }
}

printf("\nAfter destuffing the string is : %s", arr);
}
```

OUTPUT :

```
Enter number of bits :17
```

```
Enter bit string :1111101111101111
```

```
After stuffing the string is : 11111001111100111110
```

```
After destuffing the string is : 1111101111101111
```

## ii) Character Stuffing

```
#include <stdio.h>

#include <string.h>

int main()
{
    char str[50][50], frame[50][50];

    int i, j, k = 0, n;

    char flag[10];

    strcpy(flag, "g");

    char esc[10];

    strcpy(esc, "d");

    strcpy(frame[k++], "g");

    printf("\nenter the length of string: ");

    scanf("%d", &n);

    printf("\nenter string: ");

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

        gets(str[i]);

    printf("\nentered string is : ");

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

        puts(str[i]);

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

    {

        if (strcmp(str[i], flag) != 0 && strcmp(str[i], esc) != 0)
```

```

    {
        strcpy(frame[k++], str[i]);
    }
else
{
    strcpy(frame[k++], "d");
    strcpy(frame[k++], str[i]);
} }
strcpy(frame[k++], "g");
printf("\nafter character stuffing the string is :");
for (i = 0; i < k; i++)
    printf("%s", frame[i]);
} OUTPUT :
```

```

enter the length of string: 7

enter string: g
0
0
d
d
a
y

entered string is :
g
0
0
d
d
a
y

after character stuffing the string is :gdg00dddayg
-----
```

## 2) Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CRC CCIP

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

int main() {
    int i, j, keylen, msglen, flag = 0;
    char input[100], key[30], temp[30], quot[100], rem[30], key1[30];

    printf("\nEnter data: ");
    scanf("%s", input);

    printf("\nEnter key: ");
    scanf("%s", key);

    keylen = strlen(key);
    msglen = strlen(input);

    strcpy(key1, key);

    for (i = 0; i < keylen - 1; i++) {
        input[msglen + i] = '0';
    }

    for (i = 0; i < keylen; i++) {
        temp[i] = input[i];
    }

    for (i = 0; i < msglen; i++) {
        quot[i] = temp[0];

        if (quot[i] == '0') {
            for (j = 0; j < keylen; j++) {
                key[j] = '0';
            }
        } else {
            for (j = 0; j < keylen; j++) {
                key[j] = key1[j];
            }
        }
    }

    for (j = keylen - 1; j >= 0; j--) {
```



```

        if (temp[j] == key[j]) {
            rem[j - 1] = '0';
        } else {
            rem[j - 1] = '1';
        }
    }

    rem[keylen - 1] = input[i + keylen];
    strcpy(temp, rem);
}

strcpy(rem, temp);

printf("\nQuotient is: ");
for (i = 0; i < msglen; i++) {
    printf("%c", quot[i]);
}

printf("\nRemainder is: ");
for (i = 0; i < keylen - 1; i++) {
    printf("%c", rem[i]);
}

printf("\nFinal data is: ");
for (i = 0; i < msglen; i++) {
    printf("%c", input[i]);
}

for (i = 0; i < keylen - 1; i++) {
    printf("%c", rem[i]);
}

printf("\nEnter received data (final data): ");
char temp_received[20];
scanf("%s", temp_received);

for (i = 0; i < keylen; i++) {
    temp[i] = temp_received[i];
}

for (i = 0; i < msglen; i++) {
    quot[i] = temp[0];

    if (quot[i] == '0') {
        for (j = 0; j < keylen; j++) {
            key[j] = '0';
        }
    } else {

```

```

    for (j = 0; j < keylen; j++) {
        key[j] = key1[j];
    }
}

for (j = keylen - 1; j >= 0; j--) {
    if (temp[j] == key[j]) {
        rem[j - 1] = '0';
    } else {
        rem[j - 1] = '1';
    }
}

rem[keylen - 1] = temp_received[i + keylen];
strcpy(temp, rem);
}

strcpy(rem, temp);

printf("\nQuotient is: ");
for (i = 0; i < msglen; i++) {
    printf("%c", quot[i]);
}

printf("\nRemainder is: ");
for (i = 0; i < keylen - 1; i++) {
    printf("%c", rem[i]);
}

for (i = 0; i < keylen - 1; i++) {
    if (rem[i] == '0')
        flag = 0;
    else
        flag = 1;
    break;
}

if (flag == 0)
    printf("\nNo error\n");
else
    printf("\nError\n");

return 0;
}

```

OUTPUT :

```
Enter data: 1000000010000011
Enter key: 10011
Quotient is: 1001101001101000
Remainder is: 1000
Final data is: 10000000100000111000
Enter received data (final data): 10000000100000111000

Quotient is: 1001101001101000
Remainder is: 0000
No error
```

```
Enter data: 1000000010000011
Enter key: 10011
Quotient is: 1001101001101000
Remainder is: 1000
Final data is: 10000000100000111000
Enter received data (final data): 11000000100000111000

Quotient is: 1101011100010000
Remainder is: 1000
Error
```

3) Develop a simple data link layer that performs the flow control using the sliding window protocol, and Loss recovery using G0-Back-N mechanism

i) Single bit sliding window

```
#include <stdio.h>

#include <time.h>

#include <stdlib.h>

typedef struct
{
    int no;
    int ack;
} packet;

int main()
{
    int n, i;

    printf("\n1 -bit Sliding Window ");

    printf("\nEnter the number of packets to be send :");

    scanf("%d", &n);

    packet pkt[n];

    for (i = 0; i < n; i++)
    {
        pkt[i].no = i + 1;

        pkt[i].ack = 0;
```

```

    }

    srand(time(NULL));

    int x = rand() % n;

    pkt[x].no = -1;

    pkt[x].ack = 0;

    i = 0;

    while (i < n)
    {
        printf("%d\t", pkt[i].no);

        if (pkt[i].no == -1 && pkt[i].ack == 0)
        {
            pkt[i].no = i + 1;

            pkt[i].ack = pkt[i - 1].no;

            printf("\nRetransmit %d packet: \n", pkt[i].no);

            continue;
        }

        pkt[i].ack = (i > 0) ? pkt[i - 1].no : 0;

        i++;
    }

    return 0 }

```

OUTPUT :

```

1 -bit Sliding Window
Enter the number of packets to be send :8
1      2      -1
Retransmit 3 packet:
3      4      5      6      7      8
=====

```

## ii) Go back N

```

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

typedef struct{
    int no;
    int ack;
}Packet;

int main(){
    int i,j,N,M,n,st=0,p=0;
    printf("Go-Back N\n");
    printf("Number of Packets to be transmitted: ");
    scanf("%d",&N);
    Packet pkt[N];
    for(i=0;i<N;i++){
        pkt[i].no=i+1;
        pkt[i].ack=0;
    }
    srand(time(NULL));
    int x=rand()%N;
    pkt[x].no=-1;
    pkt[x].ack=0;
    printf("Window Size: ");
    scanf("%d",&M);
    i=0;
    j=i+M-1;
    while(i<N && j<N){
        int k=i;
        printf("\nWindow %d\n",++p);
        while(k<=j){
            printf("%d ",pkt[k].no);
            if(pkt[k].no==-1 && pkt[k].ack==0){
                pkt[k].no=k+1;
                n=k;
                st=1;
                k++;
                continue;
            }
            pkt[k].ack=(k>=M)?pkt[k-M].no:0;
            k++;
        }
    }

```

```

    }
    if(st==1){
        printf("\nRe-Transmission from Frame %d",n+1);
    }
    i=(st==1)?n:i+M;
    st=0;
    j=((i+M-1)>N-1)?N-1:(i+M-1);
}
return 0;
}

```

OUTPUT :

```

Go-Back N
Number of Packets to be transmitted: 8
Window Size: 2

Window 1
-1 2
Re-Transmission from Frame 1
Window 2
1 2
Window 3
3 4
Window 4
5 6
Window 5
7 8

```

## iii) Selective repeat

```

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

typedef struct{
    int no;
    int ack;
}Packet;

int main(){
    int i,j,N,M,n,st=0,p=0;
    printf("Selective Repeat\n");
    printf("Number of Packets to be transmitted: ");
    scanf("%d",&N);
    Packet pkt[N];
    for(i=0;i<N;i++){
        pkt[i].no=i+1;
        pkt[i].ack=0;
    }
    srand(time(NULL));
    int x=rand()%N;
    pkt[x].no=-1;
    pkt[x].ack=-1;
    printf("Window Size: ");
    scanf("%d",&M);
    i=0;
    j=i+M-1;
    while(i<N){
        int k=i;
        printf("\nWindow %d\n",++p);
        if(st==1){
            printf("%d ",pkt[n].no);
            i--;
            j--;
            st=0;
        }
        while(k<=j){
            printf("%d ",pkt[k].no);
            if(pkt[k].no== -1 && pkt[k].ack== -1){
                pkt[k].no=k+1;
                n=k;
            }
        }
    }
}

```



```

        st=1;
        k++;
        continue;
    }
    pkt[k].ack=(k>=M)?pkt[k-M].no:0;
    k++;
}
if(st==1){
    printf("\nRe-Transmission of Frame %d",n+1);
}
i=i+M;
j=((i+M-1)>N-1)?N-1:(i+M-1);
}
return 0;
}

```

OUTPUT :

```

Selective Repeat
Number of Packets to be transmitted: 8
Window Size: 2

Window 1
-1 2
Re-Transmission of Frame 1
Window 2
1 3
Window 3
4 5
Window 4
6 7
Window 5
8

```

#### 4) Implement Dijkstra's algorithm to compute the shortest path through a network

```
#include <stdio.h>

int main()
{
    int n, i, j, k, v, min, u, cost[10][10], s[10], dist[10], path[10];

    printf("\nEnter the number of nodes :");

    scanf("%d", &n);

    printf("\nEnter cost matrix: \n");

    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            scanf("%d", &cost[i][j]);
        }
    }

    printf("\nEnter source node :");

    scanf("%d", &v);

    for (i = 1; i <= n; i++)
    {
        s[i] = 0;

        path[i] = v;

        dist[i] = cost[v][i];
    }

    dist[v] = 0;

    for (k = 2; k <= n; k++)
```

```
{  
    min = 999;  
    u = 0;  
    for (i = 1; i <= n; i++)  
    {  
        if (s[i] != 1)  
        {  
            if (min > dist[i])  
            {  
                u = i;  
                min = dist[i];  
            }  
        }  
    }  
  
    s[u] = 1;  
    for (i = 1; i <= n; i++)  
    {  
        if (s[i] != 1)  
        {  
            if (dist[i] > min + cost[u][i])  
            {  
                dist[i] = min + cost[u][i];  
                path[i] = u;  
            }  
        }  
    }  
    printf("\n\n");
```

```

printf("distance :node path\n");
for (i = 1; i <= n; i++)
{
    printf("%d :", dist[i]);
    printf("%d", i);

    j = i;
do {
    printf("-->%d", path[j]);

    u = path[j];

    j = u;
} while (u != v);

printf("\n");

```

OUTPUT :

```

enter the number of nodes :5

enter cost matrix:
0 1 3 99 99
1 0 4 1 2
3 4 0 2 99
99 1 2 0 2
99 2 99 2 0

enter source node :1

distance :node path
0 :1-->1
1 :2-->1
3 :3-->1
2 :4-->2-->1
3 :5-->2-->1
}

```

5) Implement distance vector routing algorithm for obtaining routing tables at each node.

```
#include <stdio.h>
struct node
{
    unsigned int dist[100];
    unsigned int form[100];
} rt[100];
int main()
{
    int dmat[100][100];
    int i, j, k, count = 0;
    int n;
    printf("\nEnter the no of nodes");
    scanf("%d", &n);
    printf("\nEnter the cost matrix :\n");
    for (i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            scanf("%d", &dmat[i][j]);
            dmat[i][i] = 0;
            rt[i].dist[j] = dmat[i][j];
            rt[i].form[j] = j;
        }
    }
    do
    {
        count = 0;
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                for (k = 0; k < n; k++)
                {
                    if (rt[i].dist[j] > dmat[i][k] + rt[k].dist[j])
                    {
                        rt[i].dist[j] = rt[i].dist[k] + rt[k].dist[j];
                        rt[i].form[j] = k;
                        count++;
                    }
                }
            }
        }
    } while (count != 0);
```

```

for (i = 0; i < n; i++)
{
    printf("\n\nDVR Table for router %d:\n", i + 1);
    printf(".....\n");
    printf("| Destination | Distance | Next Hop |\n");
    printf(".....\n");
    for (j = 0; j < n; j++)
    {
        printf("|    %2d    |    %2d    |    %2d    |\n", j + 1, rt[i].dist[j], rt[i].form[j] + 1);
    }
    printf(".....\n");
}
printf("\n\n"); }

```

OUTPUT :

```
enter the no of nodes 4
```

```
enter the cost matrix :
```

```

0 2 99 1
2 0 3 7
99 3 0 11
1 7 11 0

```

```
DVR Table for router 1:
```

-----			
Destination	Distance	Next Hop	
-----			
1	0	1	
2	2	2	
3	5	2	
4	1	4	
-----			

```
DVR Table for router 2:
```

-----			
Destination	Distance	Next Hop	
-----			
1	2	1	
2	0	2	
3	3	3	
4	3	1	
-----			

DVR Table for router 3:

Destination	Distance	Next Hop
1	5	2
2	3	2
3	0	3
4	6	2

DVR Table for router 4:

Destination	Distance	Next Hop
1	1	1
2	3	1
3	6	1
4	0	4

## 6) Write a program for congestion control using Leaky bucket algorithm

```
#include <stdio.h>

int main()
{
    int b, r, n;
    int bi = 0, pl = 0, pt = 0, t = 0;
    printf("\nEnter buffer size: ");
    scanf("%d", &b);
    printf("\nEnter output transmission(leak) rate: ");
    scanf("%d", &r);
    do
    {
        pl = 0;
        bi = bi - pt;
        printf("\nEnter input transmission rate ");
        scanf("%d", &n);
        if ((bi + n) >= b)
        {
            pl = pl + (n - b);
            bi = b;
        }
        else
        {
            bi = bi + n;
        }
        printf("\nTime: %d ", t);
        printf("\nPacket loss: %d ", pl);
        printf("\nBuffer Status: %d", bi);
        printf("\nPacket transmitted: %d", pt);
        if (bi >= r)
        {
            pt = r;
        }
        else
        {
            pt = bi;
        }
    }
```



```
    printf("\n");  
    t++;  
} while (n >= 0);  
return 0;  
}
```

OUTPUT :

```
Enter buffer size: 250  
  
Enter output transmission(leak) rate: 100  
  
Enter input transmission rate 150  
  
Time: 0  
Packet loss: 0  
Buffer Status: 150  
Packet transmitted: 0  
  
Enter input transmission rate 100  
  
Time: 1  
Packet loss: 0  
Buffer Status: 150  
Packet transmitted: 100  
  
Enter input transmission rate 400  
  
Time: 2  
Packet loss: 150  
Buffer Status: 250  
Packet transmitted: 100
```

## 7) Write a program for IP Fragmentation

```
#include<stdio.h>
int main()
{
    int pktsize,headersize, MTU,i=1;
    printf("\n Enter pktsize  headersize MTU of sending packet:\n");
    scanf("%d%d%d",&pktsize,&headersize,&MTU);
    int numberoffragments,len,offset,MF,length;
    numberoffragments=pktsize-headersize;
    len=MTU-headersize;
    printf("ID \t Length \t Offset \t MF\n");
    while(numberoffragments>=0)
    {
        if(numberoffragments>=len)
        {
            MF=1;
            length=len;
        }
        else
        {
            MF=0;
            length=numberoffragments;
        }
        offset=len*(i-1);
        printf("%d\t%d\t%d\t%d\n",i,length,offset,MF);
        i++;
        numberoffragments=numberoffragments-len;
    }
}
```

OUTPUT :

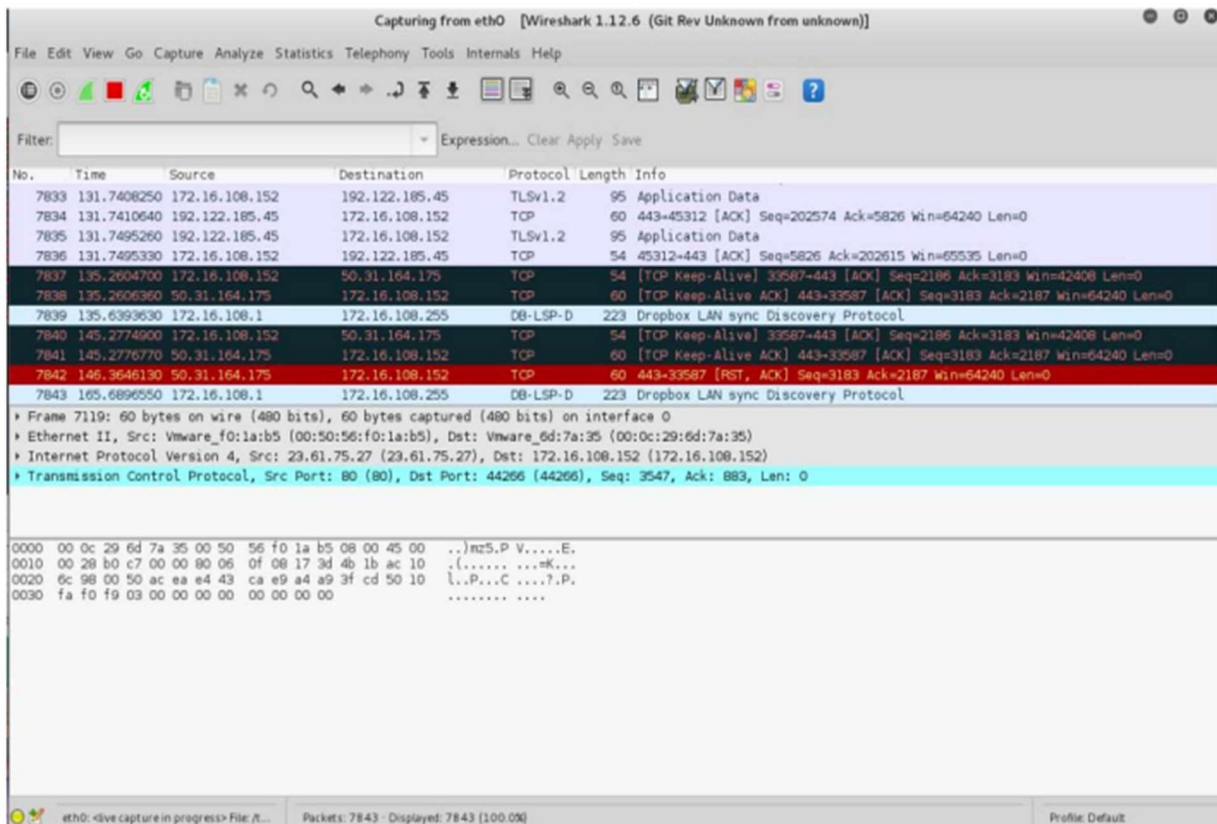
```
Enter pktsize  headersize MTU of sending packet:
4000 20 1500
ID      Length      Offset      MF
1       1480        0          1
2       1480        1480       1
3       1020        2960       0
```

## 8) Wireshark

- i) packet capturing using Wireshark
- ii) Starting Wire Shark
- iii) Viewing Captured Traffic
- iv) Analysis and Statistics & Filters

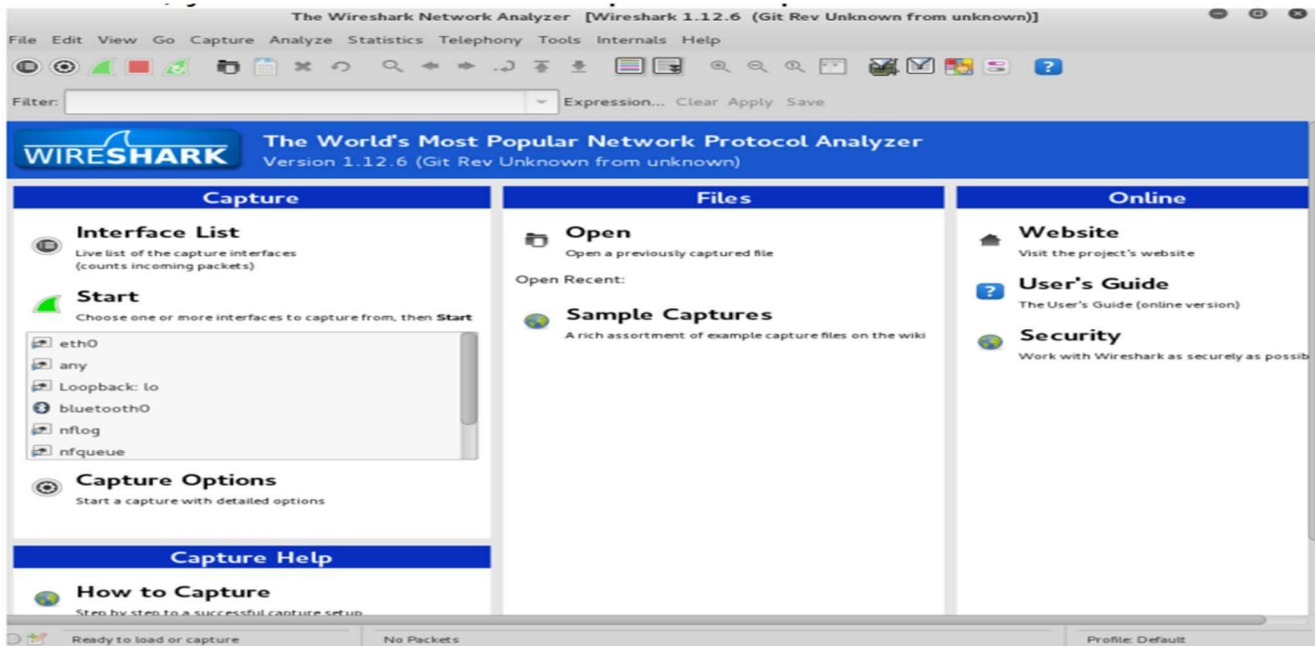
### 1) Packet Capture Using Wire shark:

After downloading and installing Wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if you want to capture traffic on the wireless network, click your wireless interface. Test Run Do the following steps: 1. Start up the Wireshark program (select an interface and press start to capture packets). 2. Start up your favorite browser (ceweasel in Kali Linux). 3. In your browser, go to Wayne State homepage by typing [www.wayne.edu](http://www.wayne.edu). 4. After your browser has displayed the <http://www.wayne.edu> page, stop Wireshark packet capture by selecting stop in the Wireshark capture window. This will cause the Wireshark capture window to disappear and the main Wireshark window to display all packets captured since you began packet capture see image below



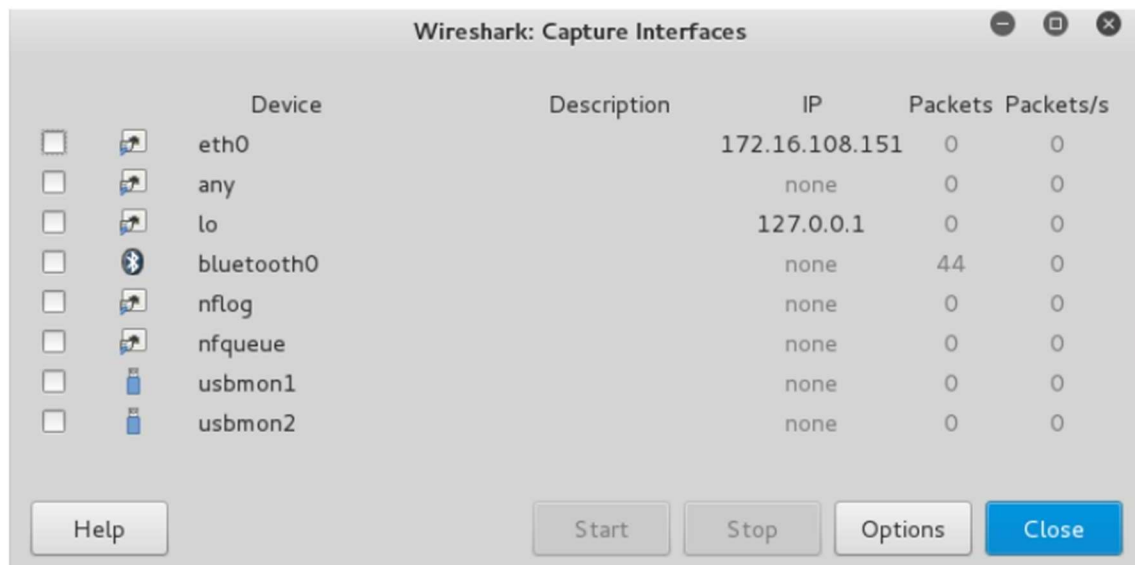
## 2) Starting Wire shark:

When you run the Wireshark program, the Wireshark graphic user interface will be shown as Figure. Currently, the program is not capturing the packets. Then, you need to choose an interface. If you are running the Wireshark on your laptop, you need to select WiFi interface. If you are at a desktop, you need to select the Ethernet interface being used. Note that there could be multiple interfaces. In general, you can select any interface but that does not mean that traffic will flow through that interface. The network interfaces (i.e., the physical connections) that your computer has to the network are shown.



**Figure 5: Initial Graphic User Interface of Wireshark**

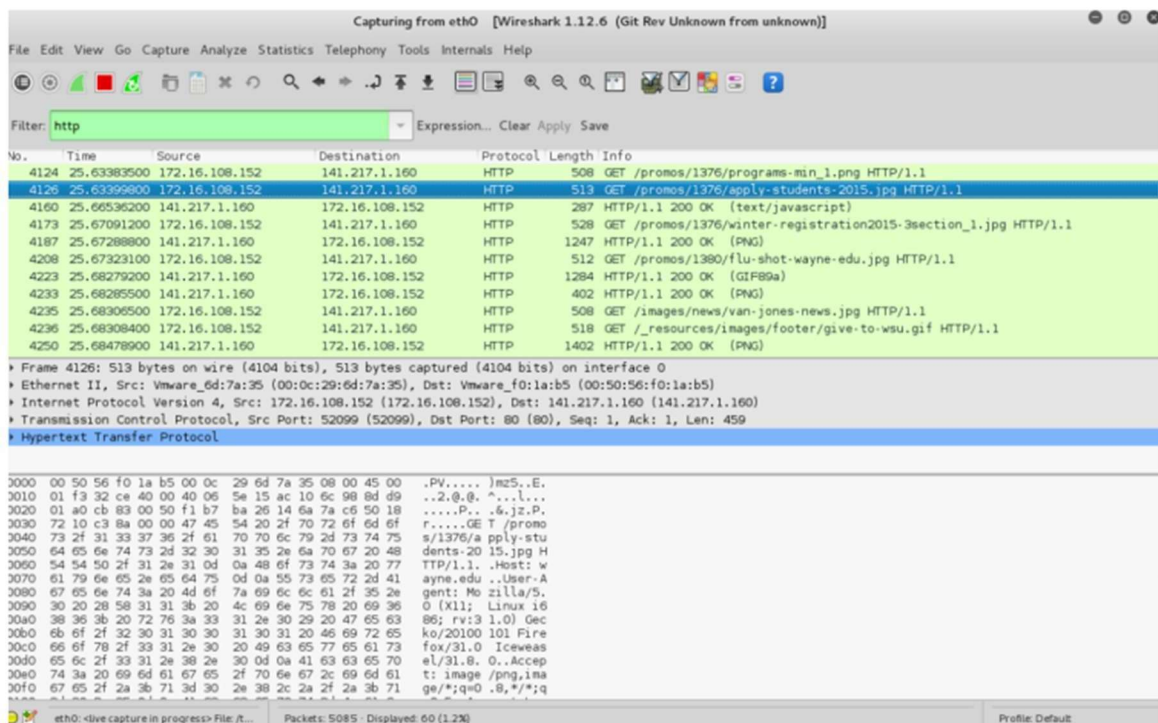
After you select the interface, you can click start to capture the packets.



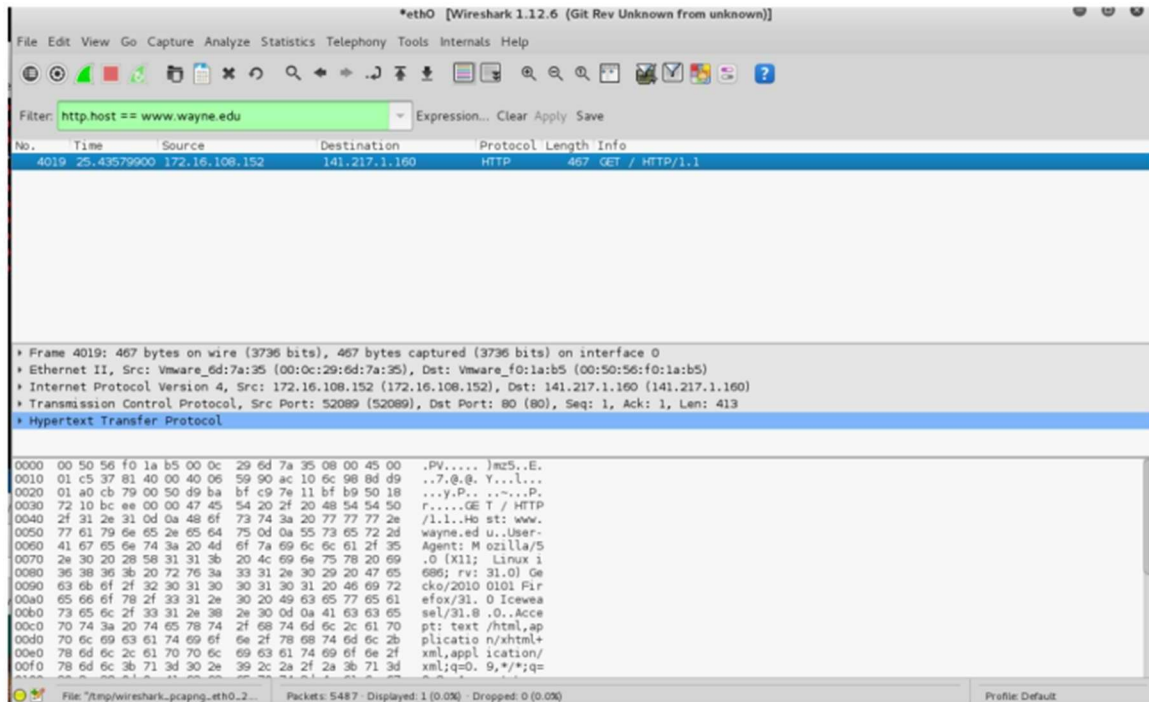
**Figure 6: Capture Interfaces in Wireshark**

### 3) Viewing Captured Traffic:

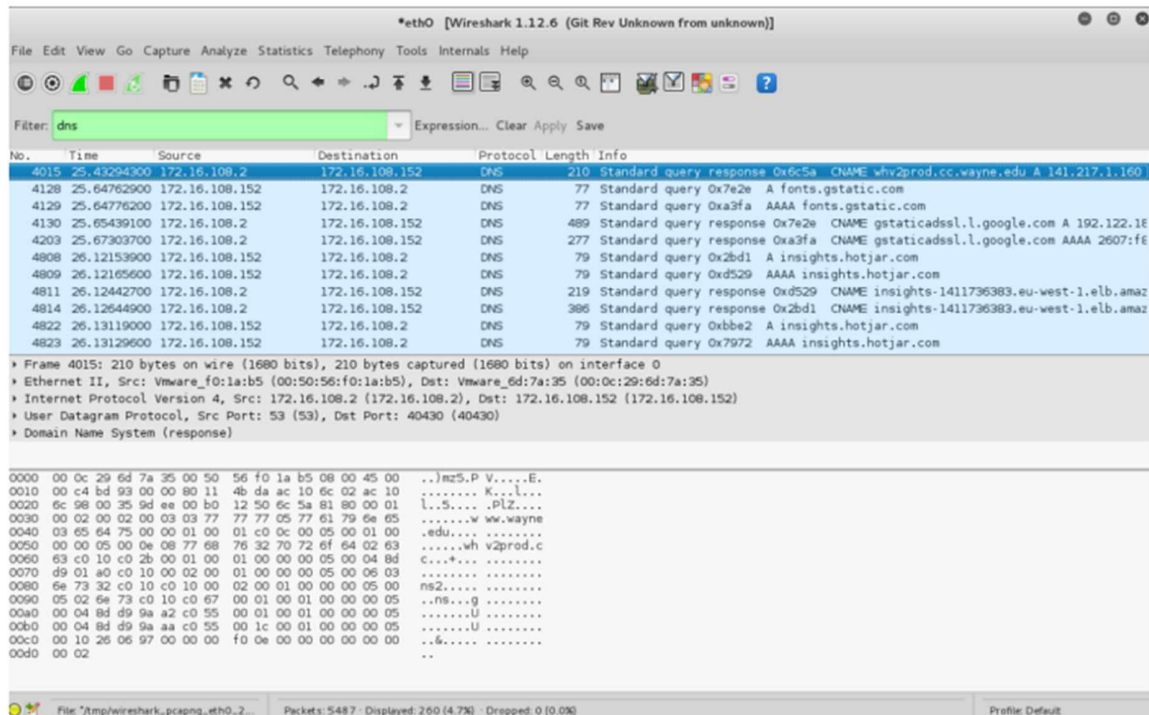
Color Coding: You'll probably see packets highlighted in green, blue, and black. Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order. You now have live packet data that contains all protocol messages exchanged between your computer and other network entities! However, as you will notice the HTTP messages are not clearly shown because there are many other packets included in the packet capture. Even though the only action you took was to open your browser, there are many other programs in your computer that communicate via the network in the background. To filter the connections to the ones we want to focus on, we have to use the filtering functionality of Wireshark by typing “http” in the filtering field as shown below:



To further filter packets in Wireshark, we need to use a more precise filter. By setting the `http.host==www.wayne.edu`, we are restricting the view to packets that have as an http host the `www.wayne.edu` website. Notice that we need two equal signs to perform the match “==” not just one. See the screenshot below



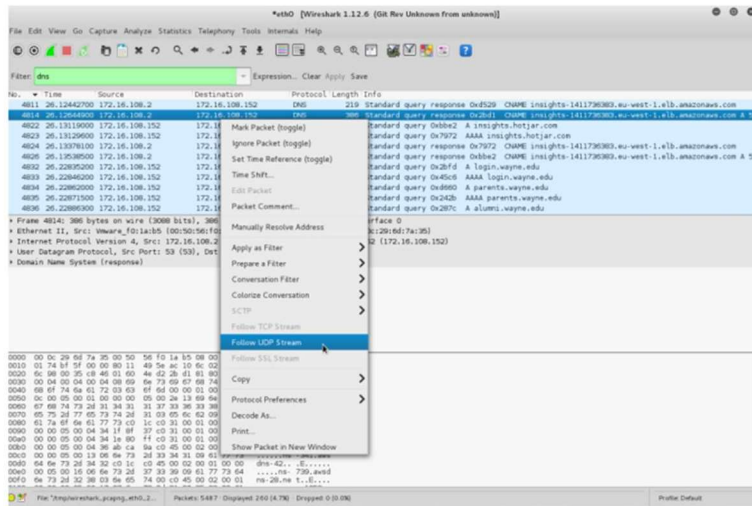
Now, we can try another protocol. Let's use Domain Name System (DNS) protocol as an example here



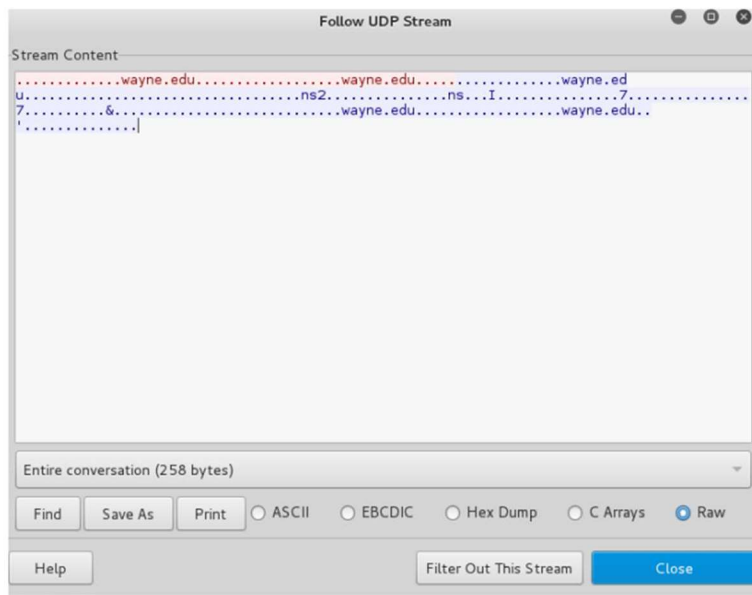


#### 4) Analysis and Statistics & Filters:

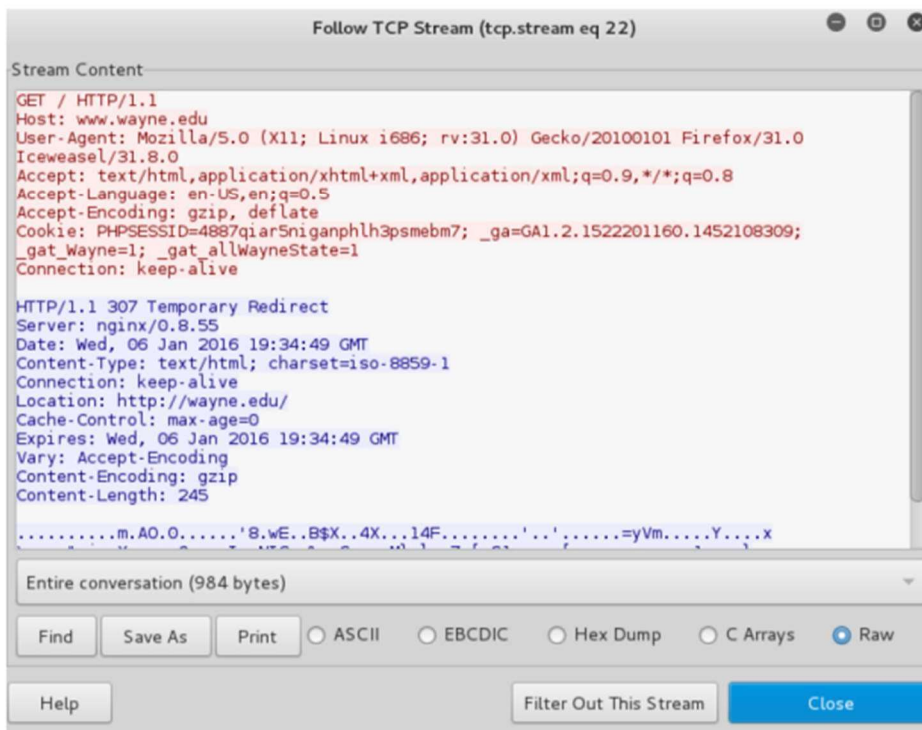
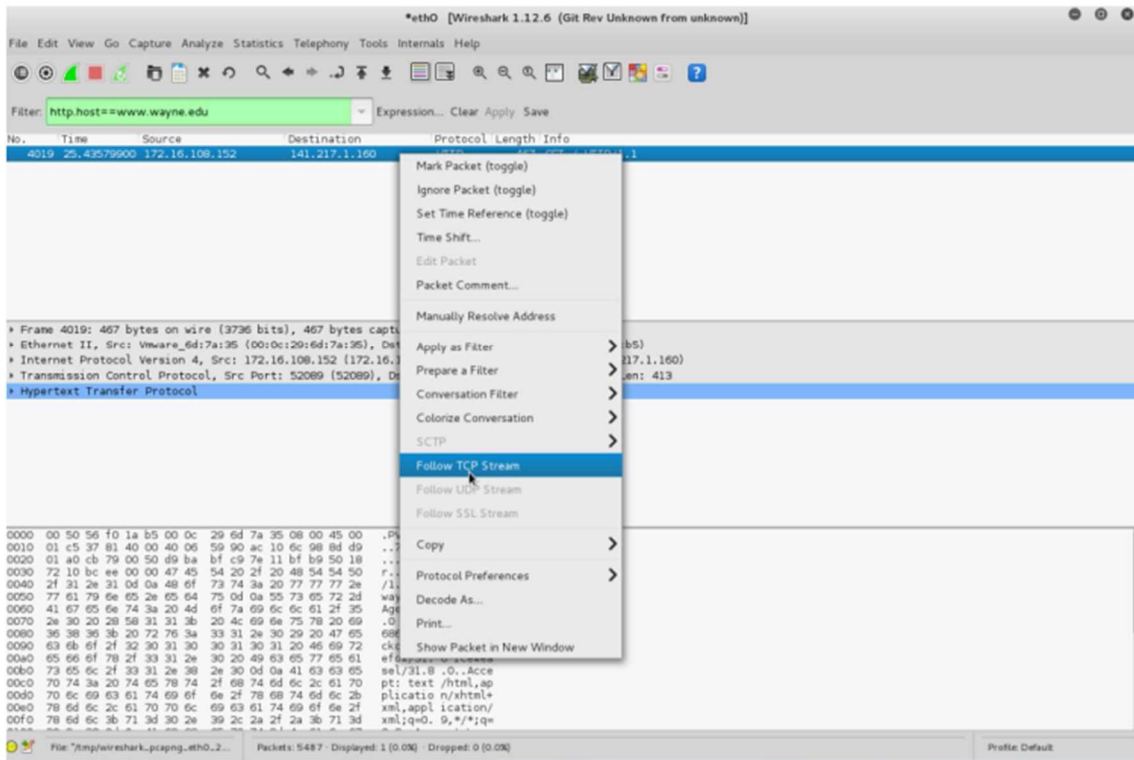
Let's try now to find out what are those packets contain by following one of the conversations (also called network flows), select one of the packets and press the right mouse button (if you are on a Mac use the command button and click), you should see something similar to the screen below



Click on Follow UDP Stream, and then you will see following screen.



If we close this window and change the filter back to “http.host==www.wayne.edu” and then follow a packet from the list of packets that match that filter, we should get something similar to the following screens. Note that we click on Follow TCP Stream this time.





9) Take a example Subnet of hosts and obtain a broadcast tree for the subnet.

//Broadcast tree for the subnet

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

```
int main()
```

```
{
```

```
    int i,j,n;
```

```
    int x,y,min;
```

```
    int cost[10][10];
```

```
    int a=0;
```

```
    int selected[10];
```

```
    printf("enter no.of nodes:");
```

```
    scanf("%d",&n);
```

```
    printf("enter costmatrix:");
```

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

```
    {
```

```
        for(j=0;j<n;j++)
```

```
        {
```

```
            scanf("%d",&cost[i][j]);
```

```
        }
```

```
    }
```

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

```
    {
```

```
        selected[i]=0;
```

```
    }
```

```
    selected[0]=1;
```

```
printf("Broadcast tree:\n");
while(a<n-1)
{
    min=99;
    for(i=0;i<n;i++)
    {
        if(selected[i]==1)
        {
            for(j=0;j<n;j++)
            {
                if(selected[j]==0)
                {
                    if(min>cost[i][j])
                    {
                        min=cost[i][j];
                        x=i;
                        y=j;
                    }
                }
            }
        }
    }
    printf("%d-->%d:%d\n",x,y,cost[x][y]);
    selected[y]=1;
    a++;
}
```

```
    return 0;  
}
```

Output:

```
enter no.of nodes:6  
enter costmatrix:  
0 2 3 99 99 99  
2 0 4 6 99 99  
3 4 0 99 1 99  
99 6 99 0 5 2  
99 99 1 5 0 3  
99 99 99 2 3 0  
Broadcast tree:  
0-->1:2  
0-->2:3  
2-->4:1  
4-->5:3  
5-->3:2
```

## 10).Implement data encryption and data decryption.

//Data encryption and decryption

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

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

```
int main()
```

```
{
```

```
    char arr[20];
```

```
    int i,c;
```

```
    printf("enter a string:\n");
```

```
    gets(arr);
```

```
    while(1)
```

```
    {
```

```
        printf("1.Encryption\n");
```

```
        printf("2.Decryption\n");
```

```
        printf("choose an option:\n");
```

```
        scanf("%d",&c);
```

```
        switch(c)
```

```
        {
```

```
            case 1:
```

```
                for(i=0;i<20 && arr[i]!='\0';i++)
```

```
                {
```

```
                    if(arr[i]==' ')
```

```
                        continue;
```

```
                    arr[i]=arr[i]+3;
```

```

    }

    printf("%s\n",arr);

    break;

case 2:

    for(i=0;i<20 && arr[i]!='\0';i++)

    {

        if(arr[i]==' ')

            continue;

        arr[i]=arr[i]-3;

    }

    printf("%s\n",arr);

    break;

case 3:

    exit(0);

default:

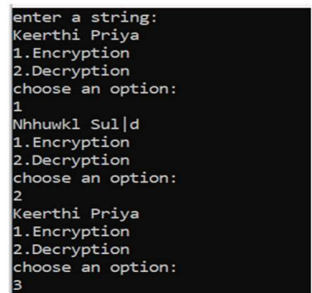
    printf("error\n");

} }

return 0;}

```

### Output:



```

enter a string:
Keerthi Priya
1.Encryption
2.Decryption
choose an option:
1
Nhhuwkl Sulld
1.Encryption
2.Decryption
choose an option:
2
Keerthi Priya
1.Encryption
2.Decryption
choose an option:
3

```

11) Write a program for frame sorting technique used in buffers.

```
//Frame sorting technique used in buffers

#include<stdio.h>

struct frame
{
    int num;
    char str[20];
};

struct frame a[10];

int n;

int main()
{
    int i,j;

    printf("enter no.of frames:\n");

    scanf("%d",&n);

    printf("enter frame seq no and its contents:\n");

    for(i=0;i<n;i++)
        scanf("%d %s",&a[i].num,&a[i].str);

    struct frame temp;

    for(i=0;i<n-1;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            if(a[j].num>a[j+1].num)
            {
```

```
        temp=a[j];
        a[j]=a[j+1];
        a[j+1]=temp;
    }
}

printf("Sorted frames are:\n");
for(i=0;i<n;i++)
    printf("%d\t%s\n",a[i].num,a[i].str);

return 0;
}
```

Output:

```
enter no.of frames:
4
enter frame seq no and its contents:
13 srija
16 vineela
14 venu
15 keerthi
Sorted frames are:
13      srija
14      venu
15      keerthi
16      vineela
```