LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING
(AUTONOMOUS)
Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified Institution
Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada
L.B.Reddy Nagar, Mylavaram-521230, Krishna Dist, Andhra Pradesh, India

COMPUTER NETWORKING LAB MANUAL

Computer Science
&ENGINEERING

NAME:______________________

ROLL NO:__________________

SEM:_______SECTION:______

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING
(AUTONOMOUS)
Accredited by NAAC with 'A' Grade, ISO 9001:2015 Certified Institution
Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada
L.B.Reddy Nagar, Mylavaram-521230, Krishna Dist, Andhra Pradesh, India

CERTIFICATE

Certified that this file is submitted by Shri/Ku. ____
Roll No.__________________a student of ______________year of the course ______________
______________________ as a part of PRACTICAL/ORALas prescribed by the Jawaharlal Nehru Technological University for the subject_______________________________________
in the laboratory of ____________________________________during the academic year
___________ and that I have instructed him/her for the said work, from time to time and I found him/her to be satisfactory progressive.

And that I have accessed the said work and I am satisfied that the same is up tothat standard envisaged for the course.

Date:- **Signature of the Faculty**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

Experiment-1

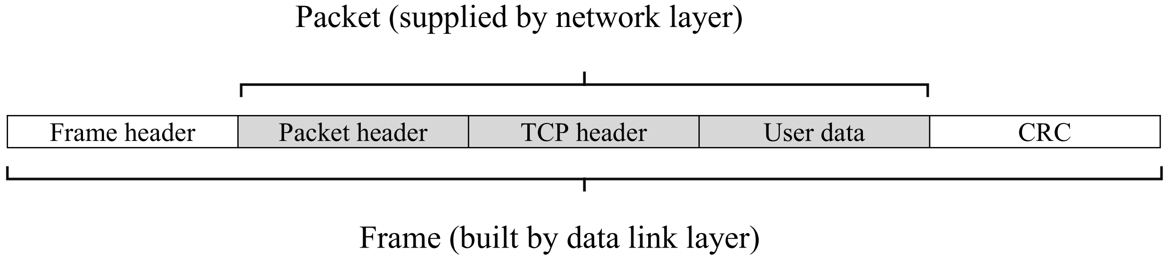
Aim: Study of Network devices in detail and connect the computers in Local Area Network

Layer	Devices	Functionality
Physical Layer	**Hub, Repeater, Modem**	A **hub** is a basic device that connects multiple devices within a network segment, broadcasting data to all connected ports. A **repeater** regenerates and amplifies network signals to extend their range and overcome distance limitations. A **modem** converts digital data from a computer into an analog signal for transmission over a line, such as a cable or phone line, and converts incoming analog signals back into digital form.
Data Link Layer	**Switch, Bridge**	* A **switch** connects devices within a local area network (LAN) and intelligently forwards data packets only to the intended recipient's port using MAC addresses. A **bridge** connects two network segments, filtering traffic and forwarding data only when it is meant for the other segment, thereby reducing network congestion.
Network Layer	**Router, Switch**	*A **router** connects multiple networks—such as a home network to the internet—and directs data packets to their destinations across these networks using logical addresses (IP addresses).
Transport Layer	**Layer 4 Switch**	*A **Layer 4 switch** can inspect packets and make forwarding or routing decisions based on Transport layer information, such as application protocols like HTTP or FTP.

(a) OSI Layers and Devices

Layer	Device/Example
Application layer	Application gateway
Transport layer	Transport gateway
Network layer	Router
Data link layer	Bridge, switch
Physical layer	Repeater, hub

(b) Encapsulation of Data


Packet (supplied by network layer)
Frame header
Packet header
TCP header
User data
CRC
Frame (built by data link layer)

Experiment-2

Aim : Write a Program to implement the data link layer farming methods such as
i) Character stuffing ii) bit stuffing.

# Character Stuffing

Source Code:

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

void main()
{
   char a[100], b[50], c[150];
   int i, j, k, m = 0, count, l1, l2;
   printf("\nEnter the data to send: ");
   fflush(stdin);
   gets(a);
   printf("\nEnter the delimiter: ");
   fflush(stdin);
   gets(b);
   l1 = strlen(a);
   l2 = strlen(b);
   c[0] = 'S';
   i = 0;
   for (k = 0; k < l2; k++)
      c[++i] = b[k];
   for (j = 0; j < l1; j++)
   {
      count = 0;
      for (k = 0, m = j; k < l2; k++, m++)
      {
         if (a[m] != b[k])
         {
            count = 1;
            break;
         }
      }

      if (count == 0)        {
         for (k = l1 + l2 - 1; k >= m; k--)
            a[k + 1] = a[k];

         for (k = 0; k < l2; k++)
            a[m++] = b[k];

         j = j + l2;
         l1 = l1 + l2;
      }
   }
   for (k = 0; k < l1; k++)
      c[++i] = a[k];
   for (k = 0; k < l2; k++)
      c[++i] = b[k];
```

```
    c[++i] = 'E'; // End character
    printf("\nData after stuffing: ");
    for (k = 0; k <= i; k++)
        printf("%c", c[k]);
    for (j = 0; j < l1 - l2; j++)
    {
        count = 0;
        m = j;
        for (k = 0; k < l2; k++)
        {
            if (a[m + k] == b[k])
                count++;
        }
        if (count == l2)
        {
            for (k = m; k < l1 - l2; k++)
                a[k] = a[k + l2];
            l1 = l1 - l2;
        }
    }
    printf("\nData after destuffing: ");
    for (i = 0; i < l1; i++)
        printf("%c", a[i]);

    printf("\n");
}
```

Output:

```
Enter the data to send: hi hello hai
Enter the delimiter: 1
Data after stuffing: S1hi hello hai1E
Data after destuffing: hi hello hai

Enter the data to send: how are you
Enter the delimiter: e
Data after stuffing: Sehow are youeE
Data after destuffing: how are you
```

#Byte Stuffing

Source Code:

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

#define FLAG_BYTE "$"
#define ESCAPE_BYTE "#"

void byte_stuff();
char input_buf[100];
char output_buf[100];

main(){
        int ans;
        do{
        input_buf[0]='\0';
        output_buf[0]='\0';
        printf("\nFLAG_BYTE:$,ESC_BYTE=#\n");
        printf("\nEnter th data from Network Layer:");
        scanf("%s",input_buf);
        byte_stuff();
        printf("\nData to the physical Layer:%s",output_buf);
        printf("\nDo you want to continue?(Y: 1/N: 0):");
        scanf("%d",&ans);
        }while(ans!=0);
return 0;
}
void byte_stuff(void){
        int i=0,j=1;
        output_buf[0]='$';
        for(;input_buf[i]!='\0';i++,j++)
        {
                if(input_buf[i]!='$' && input_buf[i]!='#')
                        output_buf[j]=input_buf[i];
                else{
                        output_buf[j++]='#';
                        output_buf[j]=input_buf[i];
                }
        }
        output_buf[j]='$';
        output_buf[j++]='\0';
}
```

Output:

```
FLAG_BYTE:$,ESC_BYTE=#
Enter the data from Network Layer:cat
Data to the physical Layer:$cat
Do you want to continue?(Y: 1/N: 0):1
FLAG_BYTE:$,ESC_BYTE=#
Enter th data from Network Layer:apple
Data to the physical Layer:$apple
Do you want to continue?(Y: 1/N: 0):0
```

//Bit stuffing

Source Code:

```
#include<stdio.h>
#include<string.h>
#define DELIM_BIT_PATTERN "01111110"
#define SNDR_INPUT 0
#define SNDR_OUTPUT 1
#define REC_INPUT 2
#define REC_OUTPUT 3

char data[4][100];

int valid_data(void);
void sender_bit_stuff(void);
void receiver_process_data(void);

int main()
{
        int ans;
        do{
        printf("\nEnter Data from Netwrok Layer in Binary Form:");
        scanf("%s",data[SNDR_INPUT]);
        if(!valid_data())
                continue;
        sender_bit_stuff();
        printf("\nSenders Physical Layer Data:%s\n",data[SNDR_OUTPUT]);
        strcpy(data[REC_INPUT],data[SNDR_OUTPUT]);
        receiver_process_data();
        printf("\nReceiver's Network Layer Data: %s\n",data[REC_OUTPUT]);
        printf("\n\nDo you want to continue?(y: 1/n: 0)");
        scanf("%d",&ans);
        }while(ans!=0);
}


int valid_data(){
        char *p=data[SNDR_INPUT];
        if(*p=='\0'){
                printf("\n***Enter Some DAta***\n");
                return 0;
        }
        while(*p!='\0'){
                if(*p!='1' && *p!='0'){
                printf("** this is not binary data. please Enter 0's and 1's\n");
                  }
                p++;
        }
return 1;
}


void sender_bit_stuff(void){
        char *src=data[SNDR_INPUT];
        char *dst=data[SNDR_OUTPUT];
        int count=0;
        strcpy(dst,DELIM_BIT_PATTERN);
        dst+=strlen(DELIM_BIT_PATTERN);
        while(*src!='\0')
        {
                if(count==5){
                        *dst='0';
                        dst+=1;
                         count=0;
                }
                if(*src=='1')
                        count++;
                else
                        count=0;
                *dst++=*src++;
        }
        if(*src=='\0' && count==5){
                *dst='0';
                dst+=1;
        }
        strcpy(dst,DELIM_BIT_PATTERN);
        dst+=strlen(DELIM_BIT_PATTERN);
        *dst='\0';
}

void receiver_process_data(void){
        char *src=data[REC_INPUT];
        char *dst=data[REC_OUTPUT];
        char *end;
        int count=0;
        src+=strlen(DELIM_BIT_PATTERN);
        end=data[REC_INPUT]+strlen(data[REC_INPUT])-strlen(DELIM_BIT_PATTERN);
        while(src<=end)
        {
                if(count==5)
                        src+=1;
                        count=0;
                if(*src=='1')
                        count++;
                else
                        count=0;
                *dst++=*src++;
        }
        *(dst-1)='\0';
return;
}
```

Output:

```
Enter Data from Netwrok Layer in Binary Form:01111111111110
Senders Physical Layer Data:0111111001111101111011001111110
Receiver's Network Layer Data: 0111110111110110

Do you want to continue?(y: 1/n: 0)0
```

Experiment-3

Aim : Write a program for Hamming Code generation for error detection and correction.

Source Code:

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

#define MAX 50

/* Function to calculate number of parity bits */
int calculateParityBits(int m) {
   int r = 0;
   while ((1 << r) < (m + r + 1)) {
      r++;
   }
   return r;
}

/* Function to position data and parity bits */
void insertDataBits(int code[], int data[], int m, int r) {
   int j = 0, i;
   for (i = 1; i <= m + r; i++) {
      if ((i & (i - 1)) == 0)
         code[i] = 0;  /* Placeholder for parity */
      else
         code[i] = data[j++];
   }
}

/* Function to set parity bits */
void setParityBits(int code[], int totalBits, int r) {
   int i, j, pos, parity;
   for (i = 0; i < r; i++) {
      pos = 1 << i;
      parity = 0;
      for (j = 1; j <= totalBits; j++) {
         if (j & pos)
            parity ^= code[j];
      }
      code[pos] = parity;
   }
}

/* Function to display code */
void printCode(int code[], int totalBits) {
   int i;
   printf("Generated Hamming Code (MSB to LSB): ");
   for (i = totalBits; i >= 1; i--) {
      printf("%d", code[i]);
   }
   printf("\n");
}
```

```
/* Function to detect error */
int detectError(int code[], int totalBits, int r) {
    int i, j, pos, parity, errorPosition = 0;
    for (i = 0; i < r; i++) {
        pos = 1 << i;
        parity = 0;
        for (j = 1; j <= totalBits; j++) {
            if (j & pos)
                parity ^= code[j];
        }
        if (parity != 0)
            errorPosition += pos;
    }
    return errorPosition;
}

void main() {
    int m, r, totalBits, i;
    int data[MAX], code[MAX];
    char ch;

    clrscr();

    printf("Enter number of data bits: ");
    scanf("%d", &m);

    printf("Enter %d data bits (from LSB to MSB):\n", m);
    for (i = m - 1; i >= 0; i--) {
        printf("Bit %d: ", m - i);
        scanf("%d", &data[i]);
    }

    r = calculateParityBits(m);
    totalBits = m + r;

    insertDataBits(code, data, m, r);
    setParityBits(code, totalBits, r);
    printCode(code, totalBits);

    printf("Do you want to introduce an error? (y/n): ");
    flushall();  /* Clear input buffer for Turbo C */
    scanf("%c", &ch);

    if (ch == 'y' || ch == 'Y') {
        int pos;
        printf("Enter bit position to flip (1 to %d): ", totalBits);
        scanf("%d", &pos);
        code[pos] ^= 1;

        printf("Received Code (with error): ");
        for (i = totalBits; i >= 1; i--) {
            printf("%d", code[i]);
        }
        printf("\n");

        int errorPos = detectError(code, totalBits, r);
        if (errorPos == 0) {
            printf("No error detected.\n");
        } else {
            printf("Error found at position: %d\n", errorPos);
            code[errorPos] ^= 1;
            printf("Corrected Code: ");
            for (i = totalBits; i >= 1; i--) {
                printf("%d", code[i]);
            }
            printf("\n");
        }
    }
 getch();
```

Output:

```
Enter number of data bits: 4
Enter 4 data bits (from LSB to MSB):
Bit 4: 1
Bit 3: 0
Bit 2: 1
Bit 1: 1
Generated Hamming Code (MSB to LSB): 1101101
Do you want to introduce an error? (y/n): y
Enter bit position to flip (1 to 7): 5
Received Code (with error): 1101001
Error found at position: 5
Corrected Code: 1101101
```

Experiment-4

Aim : Write a Program to implement data link layer farming method checksum.

Source Code:

```
#include <stdio.h>
#include <conio.h> // Required for clrscr() and getch()

int sender(int arr[], int n) {
   int checksum, sum = 0, i;
   printf("\n**** SENDER SIDE ****\n");

   for (i = 0; i < n; i++)
      sum += arr[i];

   printf("SUM IS: %d", sum);

   checksum = ~sum; // 1's complement of sum
   printf("\nCHECKSUM IS: %d", checksum);

   return checksum;
}

void receiver(int arr[], int n, int sch) {
   int checksum, sum = 0, i;

   printf("\n\n**** RECEIVER SIDE ****\n");

   for (i = 0; i < n; i++)
      sum += arr[i];

   printf("SUM IS: %d", sum);

   sum = sum + sch;
   checksum = ~sum; // 1's complement of sum

   printf("\nCHECKSUM IS: %d", checksum);

   if (checksum == 0)
      printf("\nNo Error: Data received correctly.\n");
   else
      printf("\nError Detected: Data is corrupted!\n");
}

void main() {
   int arr[10], n, i, sch;

   clrscr();

   printf("ENTER SIZE OF THE STRING (max 10): ");
   scanf("%d", &n);

   printf("ENTER THE ELEMENTS OF THE ARRAY TO CALCULATE CHECKSUM:\n");
```

```
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    sch = sender(arr, n);
    receiver(arr, n, sch);
    getch();
}
```

Output:

```
ENTER SIZE OF THE STRING (max 10): 4
ENTER THE ELEMENTS OF THE ARRAY TO CALCULATE CHECKSUM:
10
20
30
40

**** SENDER SIDE ****
SUM IS: 100
CHECKSUM IS: -101

**** RECEIVER SIDE ****
SUM IS: 100
CHECKSUM IS: 0
No Error: Data received correctly.
```

Experiment-5

Aim: Write a program to implement on a data set of characters the three CRC polynomial — CRC-12, CRC-16, and CRC-CCITT

Source Code:

```
#include <stdio.h>
#include <string.h>
void xor_Operation(char *remainder, const char *divisor) {
   int len_divisor = strlen(divisor);
   for (int i = 0; i < len_divisor; i++) {
      remainder[i] = (remainder[i] == divisor[i]) ? '0' : '1';
   }
}
void calculate_crc(char *data, const char *generator) {
   int data_len = strlen(data);
   int gen_len = strlen(generator);
   char appended_data[100];
   strcpy(appended_data, data);
   for (int i = 0; i < gen_len - 1; i++) {
      strcat(appended_data, "0");
   }
   char remainder[100];
   strncpy(remainder, appended_data, gen_len);
   remainder[gen_len] = '\0';
   for (int i = 0; i < data_len; i++) {
      if (remainder[0] == '1') {
         xor_Operation(remainder, generator);
      }
      for (int j = 0; j < gen_len - 1; j++) {
         remainder[j] = remainder[j + 1];
      }
      remainder[gen_len - 1] = appended_data[i + gen_len];
   }
   remainder[gen_len - 1] = '\0';
   printf("\nCRC Checksum: %s\n", remainder);
}

int main() {
   char data[50];
   char generator[50];
   printf("Enter data (binary string): ");
   scanf("%s", data);
   printf("Enter generator polynomial (binary string): ");
   scanf("%s", generator);
   printf("\nCalculating CRC...\n");
   calculate_crc(data, generator);
   return 0;
}
```

Output:

```
Enter data (binary string): 10110111
Enter generator polynomial: 1010
Calculating CRC...
CRC Checksum: 110
```

Aim: Program to implement Cyclic Redundancy Check CRC-12,CRC-16

Source Code:

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

char data[20];
char check_value[20];
char gen_poly[20];
int data_length;
int N;

void XOR() {
   for (int j = 0; j < N; j++) {
      check_value[j] = (check_value[j] == gen_poly[j]) ? '0' : '1';
   }
}
void crc() {
   for (int i = 0; i < data_length; i++) {
      if (check_value[0] == '1') {
         XOR();
      }
      for (int j = 0; j < N - 1; j++) {
         check_value[j] = check_value[j + 1];
      }
      check_value[N - 1] = data[i + N];
   }
}

void receiver() {
   printf("\nEnter the received data: ");
   scanf("%s", data);
   printf("\nData received: %s\n", data);
   strncpy(check_value, data, N);
   check_value[N] = '\0';
   for (int i = 0; i < data_length; i++) {
      if (check_value[0] == '1') {
         XOR();
      }
      for (int j = 0; j < N - 1; j++) {
         check_value[j] = check_value[j + 1];
      }
      check_value[N - 1] = data[i + N];
   }
   int i;
   for (i = 0; i < N - 1 && check_value[i] == '0'; i++);
   if (i == N - 1) {
      printf("\nNo error detected\n");
   } else {
      printf("\nError detected\n");
   }
}
int main() {
   printf("\nEnter data to be transmitted: ");
```

```
    scanf("%s", data);
    printf("Enter the Generating polynomial: ");
    scanf("%s", gen_poly);
    data_length = strlen(data);
    N = strlen(gen_poly);
    for (int i = data_length; i < data_length + N - 1; i++) {
        data[i] = '0';
    }
    data[data_length + N - 1] = '\0';
    strncpy(check_value, data, N);
    check_value[N] = '\0';
    printf("\nData padded with n-1 zeros: %s\n", data);
    crc();
    check_value[N - 1] = '\0';
    printf("\nCRC or check value is: %s\n", check_value);
    strcat(data, check_value);
    printf("\nFinal data to be sent: %s\n", data);
    receiver();
    return 0;
}
```

Output:

```
Enter data to be transmitted: 10110111
Enter the generating polynomial: 1010

Data padded with n-1 zeros: 10110111000

CRC or check value is: 110

Final data to be sent: 10110111110

Enter the received data: 10110111110

Data received: 10110111110

No error detected
```

Experiment-6

Aim : Write a Program to implement Sliding window protocol

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
#define MAX 50  // Maximum number of packets
int totalFrames;
int windowSize;
int frames[MAX];
int ack[MAX];  // 0 = not acknowledged, 1 = acknowledged
void receiver(int slide);
void sender() {
    int slide = 0;
    while (slide < totalFrames) {
        printf("\nSending packets in window from %d to %d:\n", slide, slide + windowSize - 1);
        for (int i = slide; i < slide + windowSize && i < totalFrames; i++) {
            if (ack[i] == 0) {
                printf("Sending packet %d\n", frames[i]);
            }
        }
        receiver(slide);
        // Slide the window if packets are acknowledged
        while (ack[slide] == 1 && slide < totalFrames) {
            slide++;
        }
    }
}

void receiver(int slide) {
    int i;
    randomize();
    for (i = slide; i < slide + windowSize && i < totalFrames; i++) {
        int random = rand() % 10;  // 0 to 9
        if (random < 8) {
            printf("Receiver got packet %d. Sending ACK.\n", frames[i]);
            ack[i] = 1;
        } else {
            printf("Packet %d lost! No ACK sent.\n", frames[i]);
            break;  // Stop: Go-Back-N requires re-sending from here
        }
    }
}
int main() {
    int i;
    clrscr();
    printf("Enter total number of packets to send: ");
    scanf("%d", &totalFrames);
    printf("Enter window size: ");
    scanf("%d", &windowSize);
```

```
   for (i = 0; i < totalFrames; i++) {
       frames[i] = i;
       ack[i] = 0;
    }
    sender();
    printf("\nAll packets sent and acknowledged!\n");
    getch();
    return 0;
}
```

Output:

```
Enter total number of packets to send: 5
Enter window size: 3
Sending packets in window from 0 to 2:
Sending packet 0
Sending packet 1
Sending packet 2
Receiver got packet 0. Sending ACK.
Receiver got packet 1. Sending ACK.
Packet 2 lost! No ACK sent.
--------------------------------------------------
// Window slides only up to the first unacknowledged packet (Packet 2)
// slide is now 2 (index of Packet 2)

Sending packets in window from 2 to 4:
Sending packet 2
Sending packet 3
Sending packet 4
Receiver got packet 2. Sending ACK.
Receiver got packet 3. Sending ACK.
Receiver got packet 4. Sending ACK.
--------------------------------------------------
// Window slides because Packet 2, 3, and 4 are now acknowledged
// slide becomes 5 (totalFrames)

All packets sent and acknowledged!
```

Experiment-7

Aim: Write a Program to implement Stop and Wait Protocol.

Source Code:

```
#include<stdlib.h>
#include<stdio.h>
#include<time.h>
#define Total_Frames 5
int frames[Total_Frames];
int receiver(int frame)
{
int ack;
ack=rand()%2;
if(ack==0)
{
printf("receiver:Frame %d lost! No ack sent \n",frame);
return 0;
}
else
{
printf("Receiver:Frame %d received sending ack------\n",frame);
return 1;
}
}
void sender()
{
int i=0;
int ackReceived;
while(i<Total_Frames)
{
printf("\n Sender:sending Frame %d\n",frames[i]);
ackReceived=receiver(frames[i]);
if(ackReceived)
{
printf("Sender:Ack for frame %d received \n",frames[i]);
i++;
}
else
{
printf("sender:Timeout! Resending Frame %d\n",frames[i]);
}
}
printf("\n All frames sent successfully!\n");
}
int main()
{
int i;
clrscr();
srand(time(0));

for(i=0;i<Total_Frames;i++)
{
frames[i]=i+1;
}
sender();
getch();
return 0;
}
```

Output:

```
Sender:sending Frame 1
Receiver:Frame 1 received sending ack------
Sender:Ack for frame 1 received

 Sender:sending Frame 2
Receiver:Frame 2 received sending ack------
Sender:Ack for frame 2 received

 Sender:sending Frame 3
receiver:Frame 3 lost! No ack sent
sender:Timeout! Resending Frame 3

 Sender:sending Frame 3
Receiver:Frame 3 received sending ack------
Sender:Ack for frame 3 received

 Sender:sending Frame 4
Receiver:Frame 4 received sending ack------
Sender:Ack for frame 4 received

 Sender:sending Frame 5
Receiver:Frame 5 received sending ack------
Sender:Ack for frame 5 received

 All frames sent successfully!
```

Experiment-8

Aim : Write a program for congestion control using leaky bucket algorithm

Source Code:

```
#include <stdio.h>
#define MIN(a,b) ((a>b)?b:a)
int main()
{
    int bucketSize, outputRate;
    int packets[10] = {0}, i = 0, n, remaining = 0, dropped = 0, extra;
    clrscr();
    // Input
    printf("Enter bucket size: ");
    scanf("%d", &bucketSize);

    printf("Enter output rate: ");
    scanf("%d", &outputRate);

    printf("Enter number of seconds: ");
    scanf("%d", &n);

    // Packets arriving per second
    for (i = 0; i < n; i++) {
        printf("Packets arriving at second %d: ", i + 1);
        scanf("%d", &packets[i]);
    }

    // Display header
    printf("\nTime\tReceived\tSent\tDropped\tRemaining\n");

    // Process each second
    for (i = 0; i < n || remaining > 0; i++) {
        int recv = (i < n) ? packets[i] : 0;

        int sent = MIN(recv + remaining, outputRate);

        extra = recv + remaining - outputRate;

        if (extra > 0) {
            if (extra > bucketSize) {
                dropped = extra - bucketSize;
                remaining = bucketSize;
            } else {
                dropped = 0;
                remaining = extra;
            }
        } else {
            dropped = 0;
            remaining = 0;
        }

        printf("%d\t%d\t\t%d\t%d\t%d\n", i + 1, recv, sent, dropped, remaining);
    }

getch();
return 0;
}
```

Output:

```
Enter bucket size: 10
Enter output rate: 4
Enter number of seconds: 5
Packets arriving at second 1: 10
Packets arriving at second 2: 8
Packets arriving at second 3: 3
Packets arriving at second 4: 0
Packets arriving at second 5: 2

Time   Received      Sent   Dropped     Remaining
1      10            4            6           4
2      8             4            2           8
3      3             4            1           7
4      0             4            0           3
5      2             4            0           1
6      0             1            0           0
```

Experiment-9

Aim: Write a Program to implement Distance vector routing algorithm by obtaining routing table at each node (Take an example subnet graph with weights indicating delay between nodes)

Source Code:

```
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}
rt[10];

int main()
{
int dmat[20][20];
int n,i,j,k,count=0;
printf("enter the number of nodes:");
scanf("%d",&n);
printf("enter the cost matrix :\n");

for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
scanf("%d",&dmat[i][j]);
dmat[i][i]=0;
rt[i].dist[j]=dmat[i][j];
rt[i].from[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].from[j]=k;
count++;
}

}while(count!=0);

for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
printf("\n state value for router %d is\n",i+1);
for(j=0;j<n;j++)
{
printf("\n node %d via %d Distance %d", j+1,rt[i].from[j]+1,rt[i].dist[j]);
}
}
printf("\n");
}
```

Output:

```
enter the number of nodes:3
enter the root matrix:
0 2 4
2 0 5
4 5 0
state value for router 1 is

node 1 via 1 distance 0
node 2 via 2 distance 2
node 3 via 3 distance 4

state value for router 2 is

node 1 via 1 distance 2
node 2 via 2 distance 0
node 3 via 3 distance 5

state value for router 3 is

node 1 via 1 distance 4
node 2 via 2 distance 5
node 3 via 3 distance 0
```

Aim: Program on Dijkshtra Algrorithm

Source Code:

```
#include <limits.h>
#include <stdio.h>
#define V 9
int minDistance(int dist[], bool sptSet[])
{
 int min = INT_MAX, min_index;
 for (int v = 0; v < V; v++)
 if (sptSet[v] == false &&dist[v] <= min)
 min = dist[v], min_index = v;
 return min_index;
}
void printSolution(int dist[], int n)
{
printf("Vertex Distance from Source\n");
 for (int i = 0; i< V; i++)
printf("%d \t %d\n", i, dist[i]);
}
void dijkstra(int graph[V][V], int src) \
{
 int dist[V];
 bool sptSet[V];
 for (int i = 0; i< V; i++)
dist[i] = INT_MAX, sptSet[i] = false;
dist[src] = 0;
 for (int count = 0; count < V - 1; count++)
 {
 int u = minDistance(dist, sptSet);
sptSet[u] = true;
 for (int v = 0; v < V; v++)
if (!sptSet[v] && graph[u][v] &&dist[u] != INT_MAX &&dist[u] + graph[u][v]
<dist[v]) dist[v] = dist[u] + graph[u][v];
 }
printSolution(dist, V);
}
int main() {
 int graph[V][V] = { { 0, 6, 0, 0, 0, 0, 0, 8, 0 },
 { 6, 0, 8, 0, 0, 0, 0, 13, 0 },
 { 0, 8, 0, 7, 0, 6, 0, 0, 2 },
 { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
 { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
 { 0, 0, 6, 14, 10, 0, 2, 0,0 },
{ 0, 0, 0, 0, 0, 2, 0, 1, 6 },
 { 8, 13, 0, 0, 0, 0, 1, 0, 7 },
 { 0, 0, 2, 0, 0, 0, 6, 7, 0 }
 };
dijkstra(graph, 0);
 return 0;
}
```

Output:

```
Vertex Distance from Source
0       0
1       6
2       14
3       23
4       25
5       16
6       17
7       8
8       16
```

Experiment- 10

Aim : Packet Capture Using Wire shark

STEP-BY-STEP APPROACH FOR INSTALLATION OF 'WIRESHARK' TOOL(WHICH IS USED FOR ANALYZE NETWORK TRAFFIC)

Wireshark:
Wireshark is a popular network protocol analyzer used to capture and inspect data
packets traveling over a network. It allows you to see detailed information about each packet,
including source and destination IP addresses, protocols used, packet size, and payload data.
It is used for troubleshooting, analysis, development and education.
Installation steps:
step1:
open browser and type www.wireshark.org
step-2:
goto download option and double click on windowsx64 installer
step-3:
after the locate the downloaded file and click on it
step-4:
idntify setup file,click next to continue
step-5:
click on noted option in the license agreement
step-6:
Capturing Packets from website using wireshark:
spep-7:
choose components
1)Androiddump
2)Etwdump
3)Randpktdump
4)sshdump,ciscodump and wifidump
5)udpdump
click next
step-8:
additonal Tasks
wireshark start menu item,desktop icon,select next

step-9:
choose install location and click next
step-10:
click install
Expreiment:a
Identifing the Network Adapter List,identifining ICMP,tcp,udp protocol
communications in ethernet
1.Identify the filter box and type tcp and click enter and notedown result

2.Identify the filter box and type udp and click enter and notedown result
3.Identify the filter box and type icmp and click enter and notedown result

<u>Output:</u>

For TCP Protocols:
Time: 1.5044, Source: 172.16.13.115, Destination: 23.55.39.14, Length: (blank), Protocol: TCP

Time: 1.522, Source: 172.16.13.115, Destination: 172.16.11.169, Length: 66, Protocol: TCP
Time: 1.5.22, Source: 172.16.13.115, Destination: 172.16.11.229, Length: 66, Protocol: TCP

For UDP Protocols:
No: 7, Time: 0.392, Source: 172.16.11.121, Destination: 23.5.5.7, Length: 130, Protocol: UDP
No: 14, Time: 0.74, Source: 172.16.11.122, Destination: 172.16.255.255, Length: 110, Protocol: NBNS
No: 30, Time: 1.40, Source: 172.16.11.131, Destination: 172.16.11.255, Length: 130, Protocol: UDP

For ICMP Protocols:
No: 2703, Time: 95.0, Source: 172.16.11.159, Destination: 172.16.11.132, Length: 138
No: 3550, Time: 96.2, Source: 172.16.11.132, Destination: 9.3.3.3, Length: 128
No: 3744, Time: 97.4, Source: 172.16.11.132, Destination: 8.8.3.3, Length: 14

Experiment-11

Aim: Program to demonstrate NS3 Simultaor

Source Code:

```
// File: scratch/udp-echo.cc
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

int main()
{
   Time::SetResolution(Time::NS);

   // Enable logging for UdpEcho applications to see output on terminal
   LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
   LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);

   NodeContainer nodes;
   nodes.Create(2);

   PointToPointHelper pointToPoint;
   pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
   pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

   NetDeviceContainer devices = pointToPoint.Install(nodes);

   InternetStackHelper stack;
   stack.Install(nodes);

   Ipv4AddressHelper address;
   address.SetBase("10.1.1.0", "255.255.255.0");

   Ipv4InterfaceContainer interfaces = address.Assign(devices);

   // Setup UDP echo server on node 1
   UdpEchoServerHelper echoServer(9);
   ApplicationContainer serverApps = echoServer.Install(nodes.Get(1));
   serverApps.Start(Seconds(1.0));
   serverApps.Stop(Seconds(10.0));

   // Setup UDP echo client on node 0
   UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 9);
   echoClient.SetAttribute("MaxPackets", UintegerValue(3));
   echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
   echoClient.SetAttribute("PacketSize", UintegerValue(1024));

   ApplicationContainer clientApps = echoClient.Install(nodes.Get(0));
   clientApps.Start(Seconds(2.0));
```

```
    clientApps.Stop(Seconds(10.0));

    Simulator::Run();
    Simulator::Destroy();

    return 0;
}
```

OUTPUT:

```
ipc@ubuntu:~/Desktop/ns-allinone-3.33/ns-3.33$ ./waf --run scratch/network.cc

Waf: Entering directory `/home/ipc/Desktop/ns-allinone-3.33/ns-3.33/build'

[2533/2601] Compiling scratch/network.cc
[2561/2601] Linking build/scratch/network

Waf: Leaving directory `/home/ipc/Desktop/ns-allinone-3.33/ns-3.33/build'

Build commands will be stored in build/compile_commands.json

'build' finished successfully (3.257s)

At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
```