Output :-

① Enter 1st String : 000
Enter 2nd String : 011
Hamming distance (000, 011) : 2

② Enter 1st String : 10101
Enter 2nd String : 11110
Hamming distance (10101, 11110) : 3

Name of Experiment: Hamming Distance Date: 8/01/24 Page No: 01

Experiment No: 01 Experiment Result:

Program to find Hamming Distance

```
#include <stdio.h>
#include <iostream.h>
#include <string.h>
int main()
{
  int i, length, count=0;
  char v1[8], v2[8];
  clrscr();
  printf("\n Write a program to find Hamming Distance\n");
  printf("Enter 1st String:");
  scanf("%s", &v1);
  printf("Enter 2nd String:");
  scanf("%s", &v2);
  length = strlen(v2);
  for(i=0; i< length; i++)
  {
    if(v1[i] != v2[i])
    {
      count++;
    }
  }
  Printf("\n Hamming Distance: %d", count);
  getch();
}
```

Teacher's Signature .....

Output

① Enter data to be transmitted: 10001

Enter generating polynomial : 11001

Data padded with n-1 zeros: 100010000

CRC or check value is: 0111

Final data to be sent: 100010111

Enter the receiver data: 11100

Data received: 11100

Error Detected

Name of Experiment: Implementation of CRC Date: 08/01/24 Page No: 02

Experiment No: 02 Experiment Result:

Program for the Implementation of CRC

```
#include <stdio.h>
#include <string.h>
#define N strlen(gen_poly)
char data[28];
char check_value[28];
char gen_poly[10];
int data_length, i, j;
void XOR(){
    for(j=1; j<N; j++)
    check_value[j] = ((check_value[j]==gen_poly[j])?'0':'1');
}
void receiver(){
  printf("Enter the received data:");
  scanf("%s", data);
  printf("\n-----\n");
  printf("Data received: %s", data);
  crc();
  for(i=0;(i<N-1) && (check_value[i]!='1');i++);
  if(i<N-1)
    printf("\n Error detected \n\n");
    else
      printf("\n No error detected \n\n");
}
void crc(){
    for(i=0; i<N; i++)
```

Teacher's Signature .....

(2)

Enter data to be transmitted: 111
Enter the generating polynomial: 001

Data padded with n-1 zeroes: 11100

CRC or check value is: 10

Final data to be sent: 11110
Enter the received data: 11110

data received: 11110
No Error detected

```
    check_value[i] = data[i];
  do{
    if (check_value[0] == '1')
    XOR();
    for (j=0; j<N-1; j++)
    check_value[j] = check_value[j+1];
    check_value[j] = data[i++];
  } while(i<= data_length + N-1);
}
int main()
{
  printf("\n Enter data to be transmitted :");
  scanf("%s", data);
  printf("\n Enter the Generating polynomial :");
  scanf("%s", gen_poly);
  data_length = strlen(data);
  for(i= data_length; i< data_length + N-1; i++)
    data[i]='0';
  printf("\n ----------");
  printf("\n Data padded with n-1 zeros : %s", data);
  printf("\n ----------");
  crc();
  printf("\n CRC or check value is : %s", check_value);
  for(i= data_length; i< data_length+N-1; i++)
    data[i] = check_value[i-data_length];
  printf("\n ----------");
  printf("\n Final data to be sent : %s", data);
```

Teacher's Signature .....

```
printf("\n ----------.\n");
receiver();
return 0;
}
```

Teacher's Signature .....

Output:

① Enter a number to check parity: 7
Parity of number 7 is odd

② Enter a number to check parity:
Parity of number 6 is even

Name of Experiment: Parity Detection Date: 4/12/23 Page No: 05

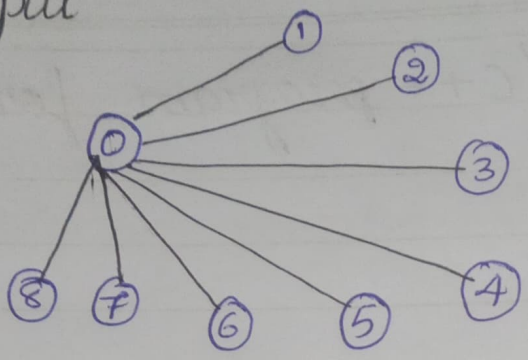
Experiment No: 03 Experiment Result:

Write and Execute a C/C++ program for Parity detection

```
#include <stdio.h>
#define bool int
bool get_parity (unsigned int n)
{
   bool parity =0;
   while(n)
   {
      parity = !parity;
      n = n & (n-1);
   }
   return parity;
}
int main()
{
   unsigned int n;
   clrscr();
   printf("Enter a number to check for parity :");
   scanf("%d", &n);
   printf("Parity of number %d = %s", n, (get_parity(n)? "odd"
                  : "even"));
   getch();
   return 0;
}
```

Teacher's Signature .....

Output



Name of Experiment: Topology Date: 11/12/23 Page No: 06

Experiment No: 04 Experiment Result:

Write and execute to simulate Network Topology

① Star Topology

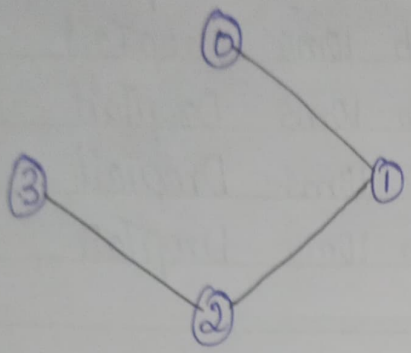
```
Set ns [new Simulator]
Set f [open out.nam w]
$ns namtrace-all $f
proc finish {} {
global ns f
$ns namtrace -all $f
close $f
exec nam out.nam &
exit 0
}
Set n0 [$ns node]
Set n1 [$ns node]
Set n2 [$ns node]
Set n3 [$ns node]
Set n4 [$ns node]
Set n5 [$ns node]
Set n6 [$ns node]
Set n7 [$ns node]
Set n8 [$ns node]

$ns duplex-link $n0 $n1 1mb 10ms DropTail
$ns duplex-link $n0 $n2 1mb 10ms DropTail
$ns duplex-link $n0 $n3 1mb 10ms DropTail
```

Teacher's Signature .....

```
$ns duplex-link $n0 $n4 1mb 10ms DropTail
$ns duplex-link $n0 $n5 1mb 10ms DropTail
$ns duplex-link $n0 $n6 1mb 10ms DropTail
$ns duplex-link $n0 $n7 1mb 10ms DropTail
$ns duplex-link $n0 $n8 1mb 10ms DropTail
```

Output

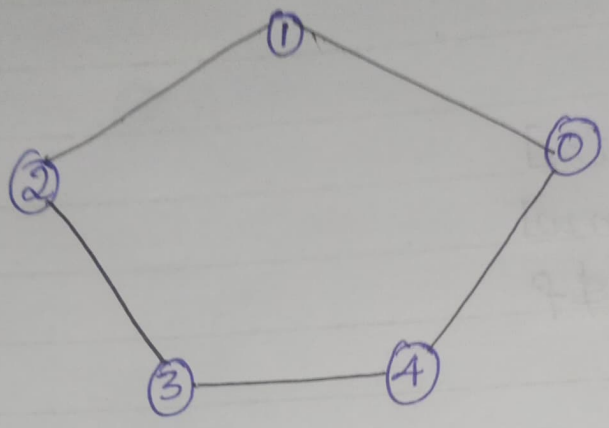


② BUS Topology

```
Set ns [new Simulator]
Set tf [open out.nam w]
$ns namtrace-all $tf
proc finish {} {
global ns tf
$ns namtrace-all $tf
close $tf
exec nam out.nam &
exit 0
}
Set n0 [$ns node]
Set n1 [$ns node]
Set n2 [$ns node]
Set n3 [$ns node]
$ns duplex-link $n0 $n1 1mb 10ms DropTail
$ns duplex-link $n1 $n2 1mb 10ms DropTail
$ns duplex-link $n2 $n3 1mb 10ms DropTail
$ns at 5.0 "finish"
$ns run
```

Teacher's Signature .....

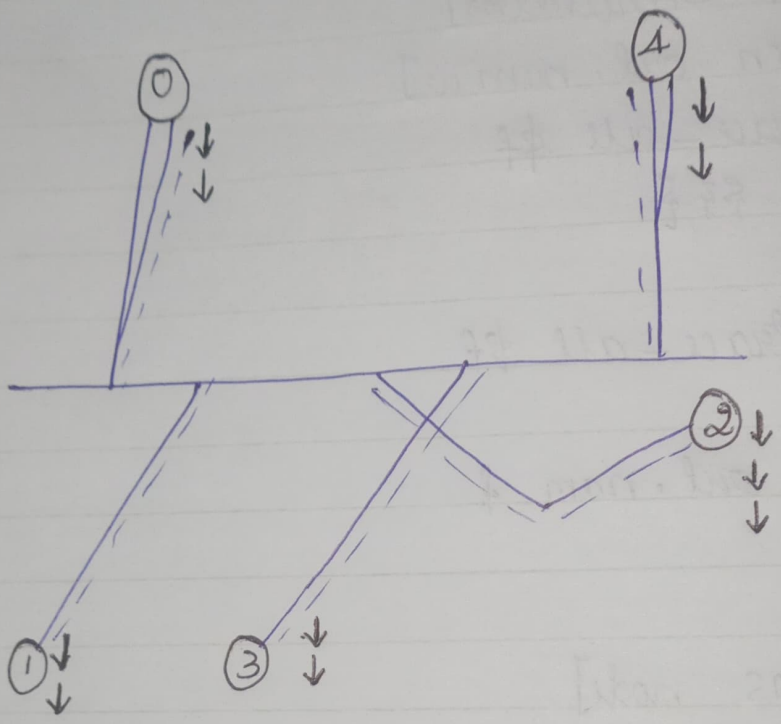
Output



⑤ Ring Topology

```
Set ns [new Simulator]
Set f [open out.nam w]
$ns namtrace-all $f
proc finish {} {
global nsf
$ns namtrace-all $f
close $f
exec nam out.nam &
exit 0
}
Set n0 [$ns node]
Set n1 [$ns node]
Set n2 [$ns node]
Set n3 [$ns node]
Set n4 [$ns node]
$ns duplex-link $n0 $n1 1mb 10ms DropTail
$ns duplex-link $n1 $n2 1mb 10ms DropTail
$ns duplex-link $n2 $n3 1mb 10ms DropTail
$ns duplex-link $n3 $n4 1mb 10ms DropTail
$ns duplex-link $n4 $n0 1mb 10ms DropTail
```

Output



Name of Experiment: BUS Protocol through Simulator Date: 01/01/24 Page No: 10

Experiment No: 05 Experiment Result:

Write and execute NS2 code to create Scenario and Study the performance of token bus protocol through simulation

```
Object set ns [new Simulator]
file set nf [open out.nam w]
$ns namtrace -all $nf

proedure proc finish {}
{
  global ns nf
  $ns flush-trace
  file close $nf
  trace file exec nam
   out.nam &
   exit 0
}
node set n0 [$ns node]
node set n1 [$ns node]
node set n2 [$ns node]
node set n3 [$ns node]
node  Set n4 [$ns node]

set lan0 ($ns new lan "$n0 $n1 $n2 $n3 $n4" 0.5Mb
 40 ms LL Queue/DropTail MAC/Csma/Cd Channel]
node n0 set tcp 0 [new Agent/TCP]
$tcp 0 set class_1
```

Teacher's Signature .....

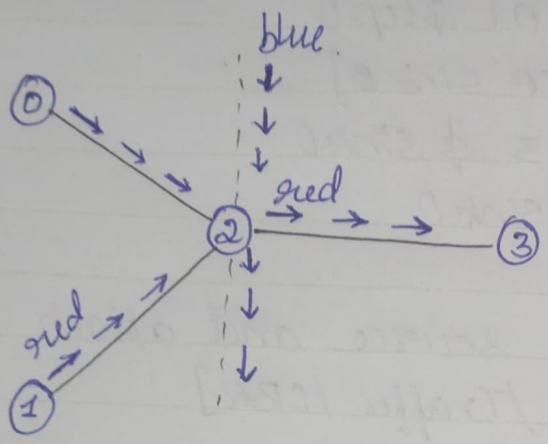
```
$ns attach-agent $n1 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0

# create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval 0.01
$cbr0 attach-agent $tcp0

$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

Teacher's Signature .....

Output:



CBR packet size = 1000
CBR interval = 0.008000000000000002

Name of Experiment..... Date 08/01/24

Experiment No 06 Experiment Result.....

Page No 12

Write and Execute a four node point-to-point network with the links connected as follows: n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and FTP between n1-n3. Apply relevant applications over TCP and FTP agents changing the parameter and determine the number of packets sent by TCP/FTP

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 2mb 10ms DropTail
$ns duplex-link $n1 $n2 2mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7mb 20ms DropTail
$ns queue-limit $n2 $n3 10
```

Teacher's Signature .....

```
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5
Set tcp [new Agent/TCP]
$tcp set class-2
$ns attach-agent $n0 $tcp
Set sink [new Agent/TCP Sink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid-1
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type-FTP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
Set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid-2
Set cbr [new Application/Traffic/CBR]
$cbr attach-agent $ndp
$cbr set type-CBR
$cbr set packet_size-1000
$cbr Set rate-1mb
$cbr Set random-false
$ns at 0.1 "$cbr start"
```

Teacher's Signature .....

```
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns
detach-agent $n3 $sink"
$ns at 5.0 "finish"
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"
$ns run.
```