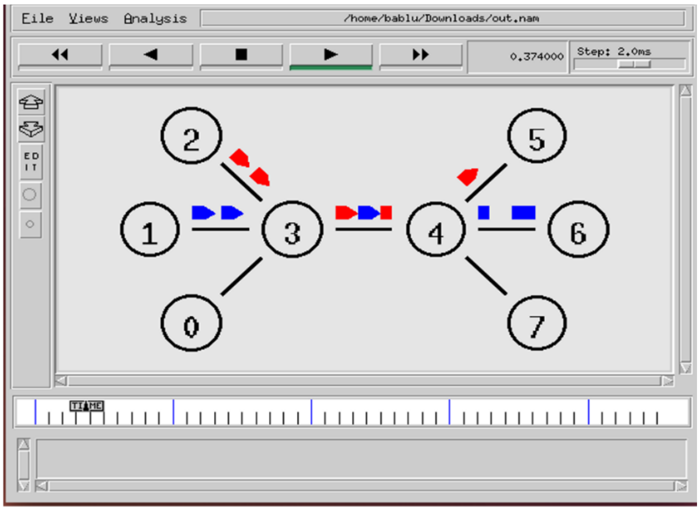
Program:

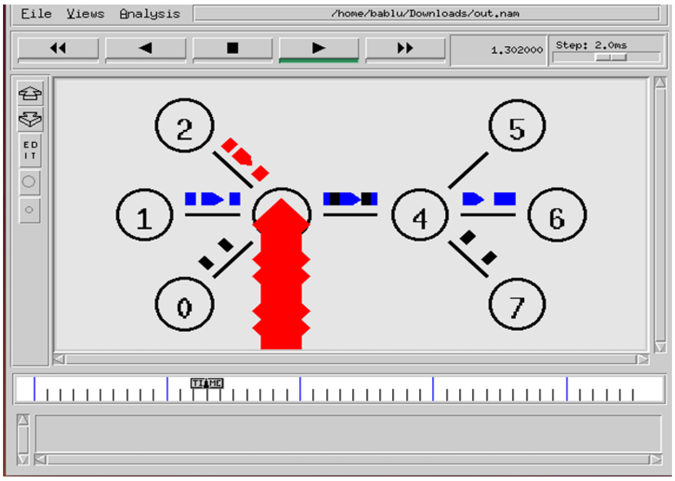
```
set ns [new Simulator]

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]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

$ns duplex-link $n0 $n3 1Mb 10ms RED

$ns duplex-link $n1 $n3 1Mb 10ms RED

$ns duplex-link $n2 $n3 1Mb 10ms RED

$ns duplex-link $n3 $n4 1Mb 10ms RED

$ns duplex-link $n4 $n5 1Mb 10ms RED

$ns duplex-link $n4 $n6 1Mb 10ms RED

$ns duplex-link $n4 $n7 1Mb 10ms RED

$ns duplex-link-op $n0 $n3 orient right-up

$ns duplex-link-op $n3 $n4 orient middle

$ns duplex-link-op $n2 $n3 orient right-down

$ns duplex-link-op $n4 $n5 orient right-up

$ns duplex-link-op $n4 $n7 orient right-down

$ns duplex-link-op $n1 $n3 orient right
```

```
$ns duplex-link-op $n6 $n4 orient left
set udp0 [new Agent/UDP]
$ns attach-agent $n2 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n5 $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null1 [new Agent/Null]
$ns attach-agent $n6 $null1
$ns connect $udp1 $null1
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2
set null2 [new Agent/Null]
$ns attach-agent $n7 $null2
$ns connect $udp2 $null2
$udp0 set fid_ 1
$udp1 set fid_ 2
$udp2 set fid_ 3
```

```
$ns color 1 Red

$ns color 2 Green

$ns color 3 Blue

$ns at 0.1 "$cbr0 start"

$ns at 0.2 "$cbr1 start"

$ns at 0.5 "$cbr2 start"

$ns at 4.0 "$cbr2 stop"

$ns at 4.2 "$cbr1 stop"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run
```

Output:





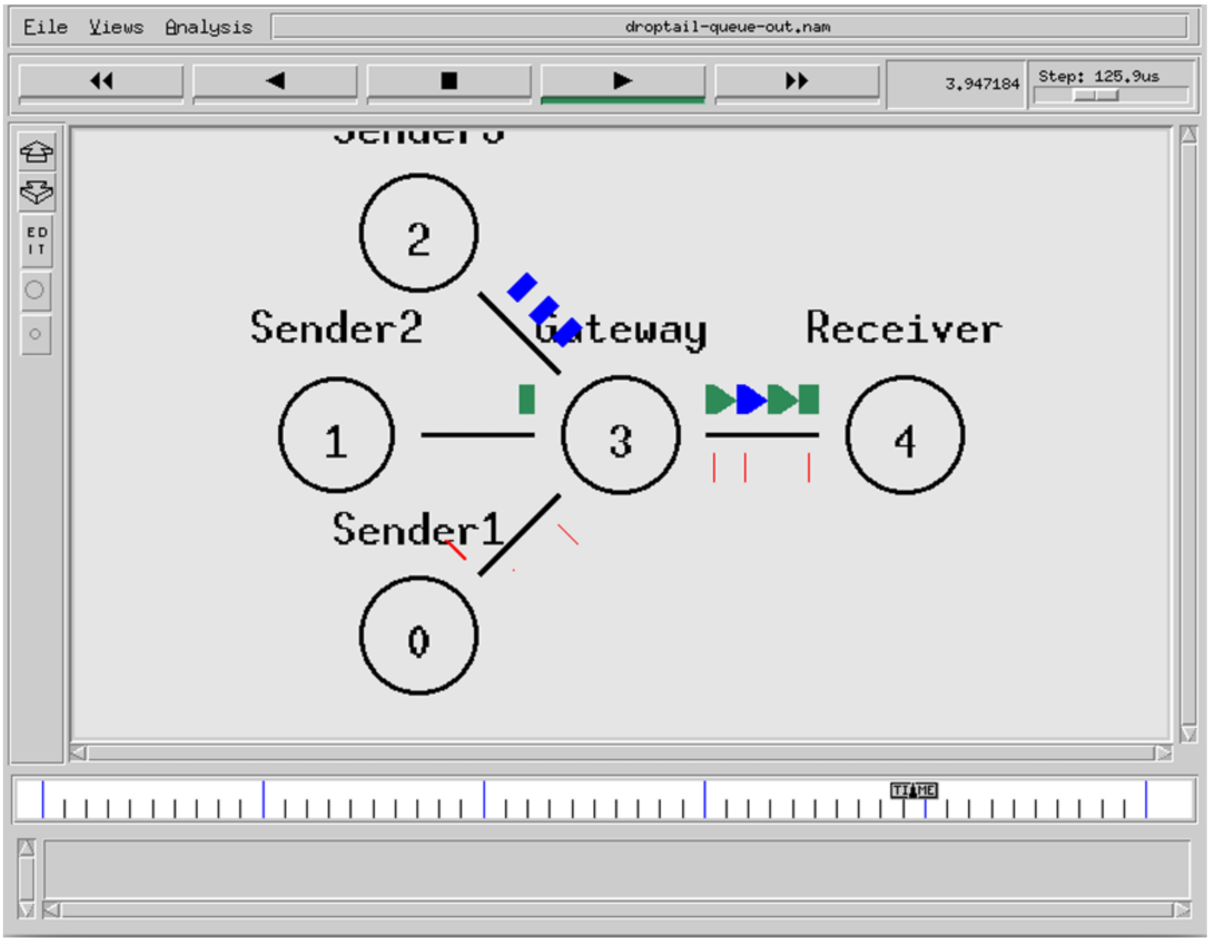
Program:

```
set ns [new Simulator]
set f [open droptail-queue-out.tr w]
$ns trace-all $f
set nf [open droptail-queue-out.nam w]
$ns namtrace-all $nf
set s1 [$ns node]
set s2 [$ns node]
set s3 [$ns node]
set G [$ns node]
set r [$ns node]
$ns color 1 red
$ns color 2 SeaGreen
$ns color 3 blue
$ns duplex-link $s1 $G 6Mb 10ms DropTail
$ns duplex-link $s2 $G 6Mb 10ms DropTail
$ns duplex-link $s3 $G 6Mb 10ms DropTail
$ns duplex-link $G $r 3Mb 10ms DropTail
$ns duplex-link-op $s1 $G orient right-up
$ns duplex-link-op $s2 $G orient right
$ns duplex-link-op $s3 $G orient right-down
$ns duplex-link-op $G $r orient right
$ns queue-limit $G $r 5
$ns duplex-link-op $s1 $G queuePos 0.5
$ns duplex-link-op $s2 $G queuePos 0.5
$ns duplex-link-op $s3 $G queuePos 0.5
$ns duplex-link-op $G $r queuePos 0.5
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $s1 $tcp1
$tcp1 set window_ 8
$tcp1 set fid_ 1
```

```
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $s2 $tcp2
$tcp2 set window_ 8
$tcp2 set fid_ 2
set tcp3 [new Agent/TCP/Reno]
$ns attach-agent $s3 $tcp3
$tcp3 set window_ 4
$tcp3 set fid_ 3
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
set sink3 [new Agent/TCPSink]
$ns attach-agent $r $sink1
$ns attach-agent $r $sink2
$ns attach-agent $r $sink3
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
$ns connect $tcp3 $sink3
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
proc finish {} {
  global ns
  $ns flush-trace
  puts "running nam..."
  exec nam -a droptail-queue-out.nam &
  exit 0
}
```

```
$ns at 0.0 "$s1 label Sender1"

$ns at 0.0 "$s2 label Sender2"

$ns at 0.0 "$s3 label Sender3"

$ns at 0.0 "$G label Gateway"

$ns at 0.0 "$r label Receiver"

$ns at 0.1 "$ftp1 start"

$ns at 0.1 "$ftp2 start"

$ns at 0.1 "$ftp3 start"

$ns at 5.0 "$ftp1 stop"

$ns at 5.0 "$ftp2 stop"

$ns at 5.0 "$ftp3 stop"

$ns at 5.25 "finish"

$ns run
```

Output:

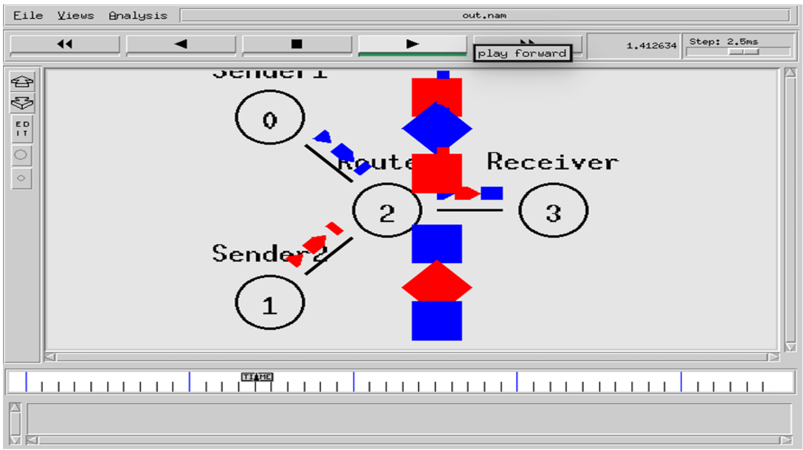


Program:

```
set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

set nf [open out.nam w]

$ns namtrace-all $nf

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

$ns duplex-link $n0 $n2 1Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n3 $n2 1Mb 10ms SFQ

$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 udp0 [new Agent/UDP]

$udp0 set class_ 1

$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0

set udp1 [new Agent/UDP]

$udp1 set class_ 2

$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize_ 500

$cbr1 set interval_ 0.005

$cbr1 attach-agent $udp1
```

```
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
$ns connect $udp1 $null0
proc finish {} {
  global ns nf
  $ns flush-trace
  close $nf
  exec nam -a out.nam &
  exit 0
}
$ns at 0.0 "$n0 label Sender1"
$ns at 0.0 "$n1 label Sender2"
$ns at 0.0 "$n2 label Router"
$ns at 0.0 "$n3 label Receiver"
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

Output:

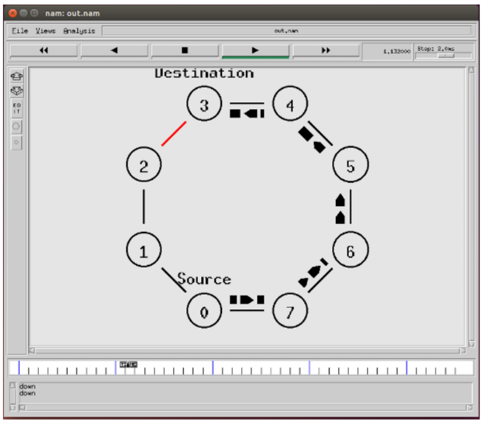


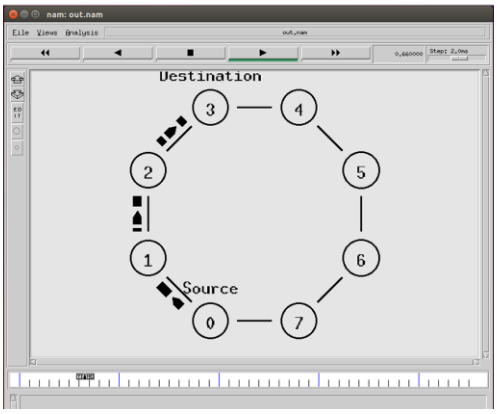
Program:

```
set ns [new Simulator]
$ns rtproto DV
set nf [open out.nam w]
$ns namtrace-all $nf
set nt [open trace.tr w]
$ns trace-all $nt
proc finish {} {
  global ns nf
  $ns flush-trace
  close $nf
  exec nam -a out.nam &
  exit 0
}
set n1 [$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 $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 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n6 1Mb 10ms DropTail
$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n1 1Mb 10ms DropTail
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
```

```
$ns duplex-link-op $n3 $n4 orient right-up

$ns duplex-link-op $n4 $n5 orient right

$ns duplex-link-op $n5 $n6 orient right-down

$ns duplex-link-op $n6 $n7 orient down

$ns duplex-link-op $n7 $n8 orient left-down

$ns duplex-link-op $n8 $n1 orient left

set udp0 [new Agent/UDP]

$ns attach-agent $n1 $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]

$ns attach-agent $n4 $null0

$ns connect $udp0 $null0

$ns at 0.0 "$n1 label Source"

$ns at 0.0 "$n4 label Destination"

$ns at 0.5 "$cbr0 start"

$ns rtmodel-at 1.0 down $n3 $n4

$ns rtmodel-at 2.0 up $n3 $n4

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run
```

Output:



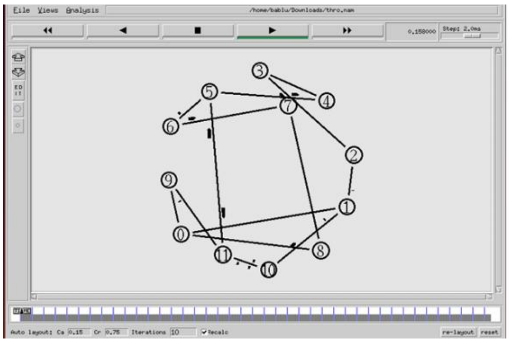


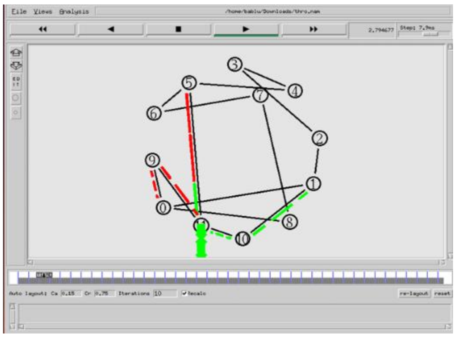
Program:

```
set ns [new Simulator]
set nr [open thro.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf
proc finish { } {
  global ns nr nf
  $ns flush-trace
  close $nf
  close $nr
  exec nam thro.nam &
  exit 0
}
for { set i 0 } { $i < 12 } { incr i 1 } {
  set n($i) [$ns node]
}
for { set i 0 } { $i < 8 } { incr i } {
  $ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail
}
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

```
set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n(1) $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize_ 500

$cbr1 set interval_ 0.005

$cbr1 attach-agent $udp1

set null1 [new Agent/Null]

$ns attach-agent $n(5) $null1

$ns connect $udp1 $null1

$ns rtproto LS

$ns rtmodel-at 10.0 down $n(11) $n(5)

$ns rtmodel-at 15.0 down $n(7) $n(6)

$ns rtmodel-at 30.0 up $n(11) $n(5)

$ns rtmodel-at 20.0 up $n(7) $n(6)

$udp0 set fid_ 1

$udp1 set fid_ 2

$ns color 1 Red

$ns color 2 Green

$ns at 1.0 "$cbr0 start"

$ns at 2.0 "$cbr1 start"

$ns at 45 "finish"

$ns run
```

Output:





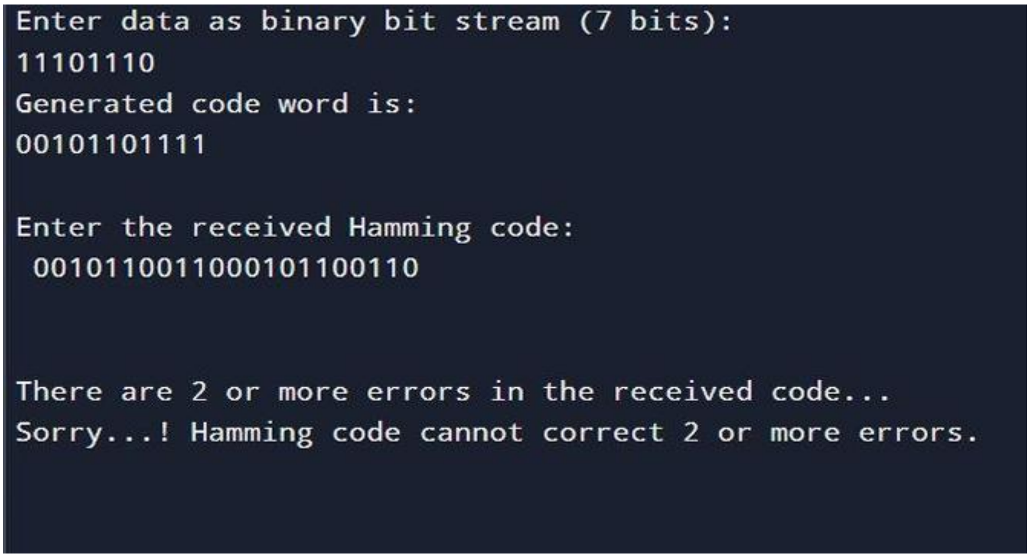
Program:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
int main() {
    int i, j, k, count, err_pos = 0, flag = 0;
    char dw[20], cw[20], data[20];
    printf("Enter data as binary bit stream (7 bits):\n");
    scanf("%s", data);
    for (i = 1, j = 0, k = 0; i < 12; i++) {
        if (i == (int)pow(2, j)) {
            dw[i] = '?';
            j++;
        } else {
            dw[i] = data[k];
            k++;
        }
    }
    for (i = 0; i < 4; i++) {
        count = 0;
        for (j = (int)pow(2, i); j < 12; j += (int)pow(2, i) * 2) {
            for (k = 0; k < (int)pow(2, i) && j + k < 12; k++) {
                if (dw[j + k] == '1') count++;
            }
        }
        dw[(int)pow(2, i)] = (count % 2 == 0) ? '0' : '1';
    }
    printf("Generated code word is:\n");
    for (i = 1; i < 12; i++) {
        printf("%c", dw[i]);
    }
```

```
printf("\n\nEnter the received Hamming code:\n");
scanf("%s", cw);
for (i = 12; i > 0; i--) {
  cw[i] = cw[i - 1];
}
for (i = 0; i < 4; i++) {
  count = 0;
  for (j = (int)pow(2, i); j < 12; j += (int)pow(2, i) * 2) {
    for (k = 0; k < (int)pow(2, i) && j + k < 12; k++) {
      if (cw[j + k] == '1') count++;
    }
  }
  if (count % 2 != 0) {
    err_pos = err_pos + (int)pow(2, i);
  }
}
if (err_pos == 0) {
  printf("\n\nThere is no error in the received code word.\n");
} else {
  if (cw[err_pos] == dw[err_pos]) {
    printf("\n\nThere are 2 or more errors in the received code...\n");
    printf("Sorry...! Hamming code cannot correct 2 or more errors.\n");
    flag = 1;
  } else {
    printf("\n\nThere is an error in bit position %d of the received code word.\n", err_pos);
    if (flag == 0) {
      cw[err_pos] = (cw[err_pos] == '1') ? '0' : '1';
      printf("\n\nCorrected code word is:\n");
      for (i = 1; i < 12; i++) {
        printf("%c", cw[i]);
      }
    }
```

```
      }
    }
    printf("\n\n");
    return 0;
}
```

Output:



```
Enter data as binary bit stream (7 bits):
11101110
Generated code word is:
00101101111

Enter the received Hamming code:
 0010110011000101100110


There are 2 or more errors in the received code...
Sorry...! Hamming code cannot correct 2 or more errors.
```