

SESSION 2022-23

CN

(Computer Networks)

Lab File

**COURSE:- BCA**

**ROLL NO :- 41221139**

**SUBMITTED BY :- SUBMITTED TO:-**

**Sachin Rajbhar Ms. Bushra Jamal**

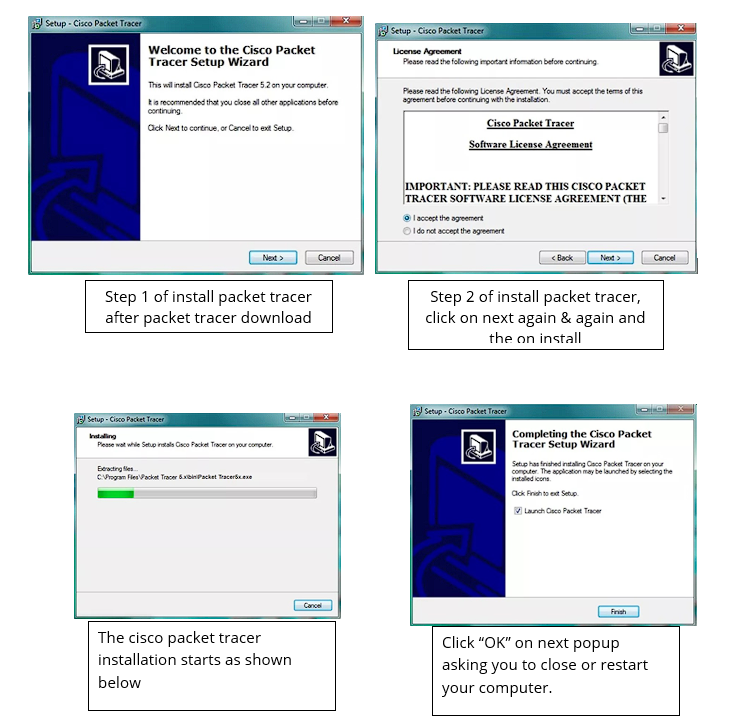
**Index**

|  |  |  |
| --- | --- | --- |
| *S.No* | *Title* | *Sign* |
| 1 | To Install Cisco Packet Tracer |  |
| 2 | To perform Message Transfer |  |
| 3 | To perform Bus Topology |  |
| 4 | To perform Ring Topology |  |
| 5 | To differentiate b/w hub & switch |  |
| 6 | To perform Star Topology |  |
| 7 | To perform Mesh Topology |  |
| 8 | To perform Hybrid Topology (Bus+Ring) |  |
| 9 | To perform Message Transfer B/W Two Switches with Router |  |
| 10 | To perform python visualization (pythontutor.com) |  |
| 11 | To perform Implementation of Data link framing method - Bit stuffing and Destuffing in python/c/c++.. |  |
| 12 | Implementation of Error detection method - even and odd parity. |  |
| 13 | To perform Static and Dynamic Routing |  |

**PRACTICAL 1**

**AIM**: To Install Cisco Packet Tracer

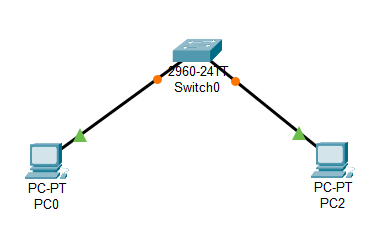
**DIAGRAM:**

****

**PRACTICAL 2**

**AIM**: To perform Message Transfer

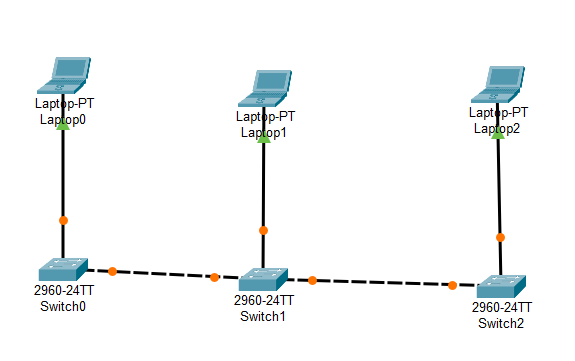
**DIAGRAM:**

****

**PRACTICAL 3**

**AIM**: To perform Bus Topology

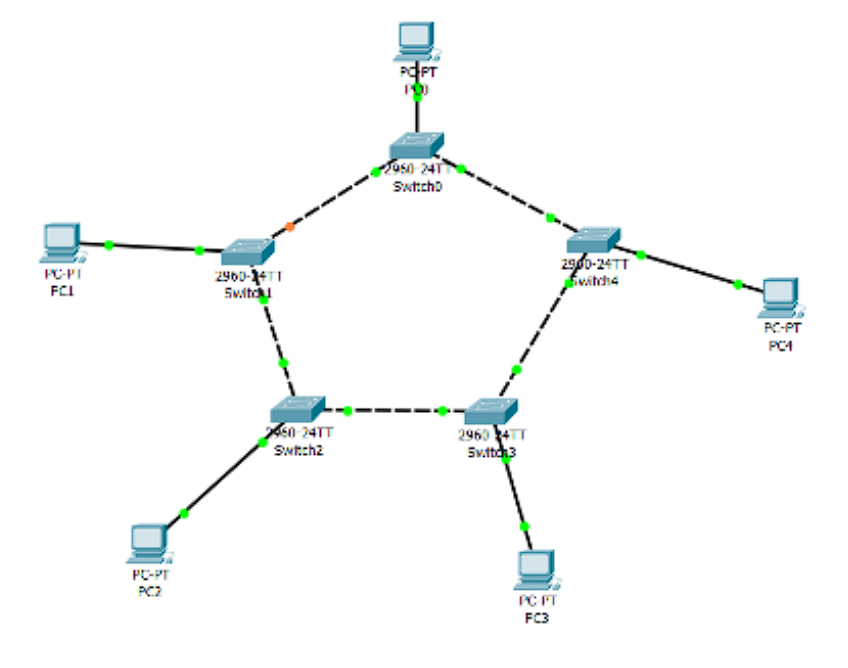
**DIAGRAM:**

****

**PRACTICAL 4**

**AIM**: To perform Ring Topology

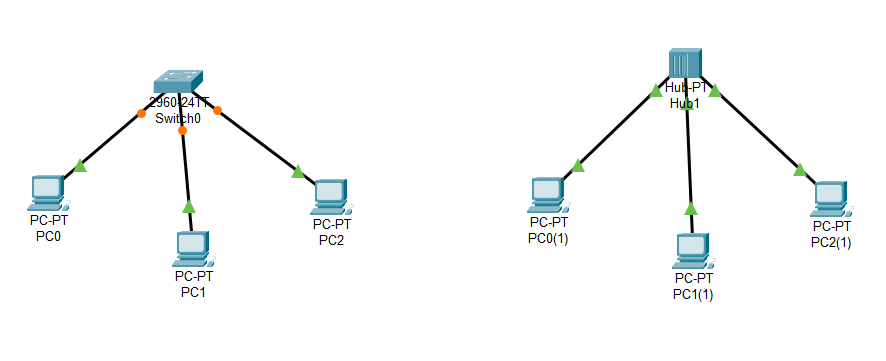
**DIAGRAM:**

****

**PRACTICAL 5**

**AIM**: To differentiate b/w hub & switch

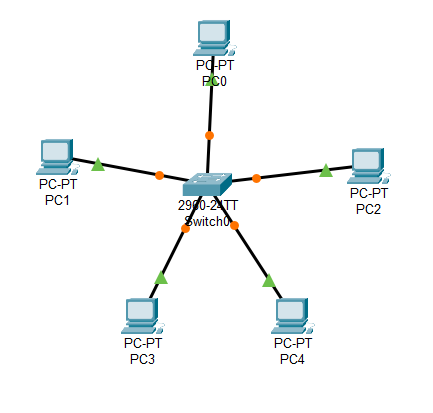
**DIAGRAM:**

****

**PRACTICAL 6**

**AIM**: To perform Star Topology

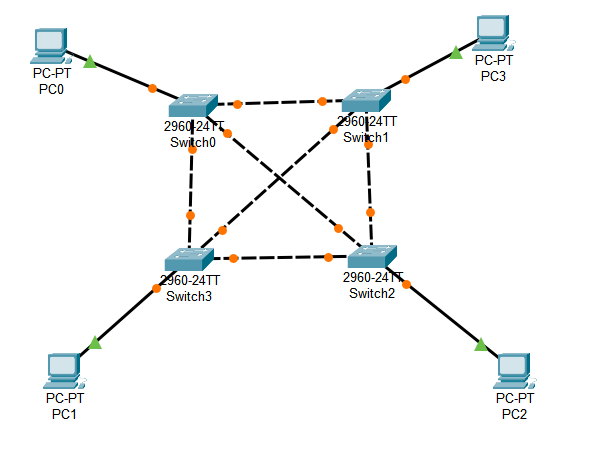
**DIAGRAM:**

**c**

**PRACTICAL 7**

**AIM**: To perform Mesh Topology

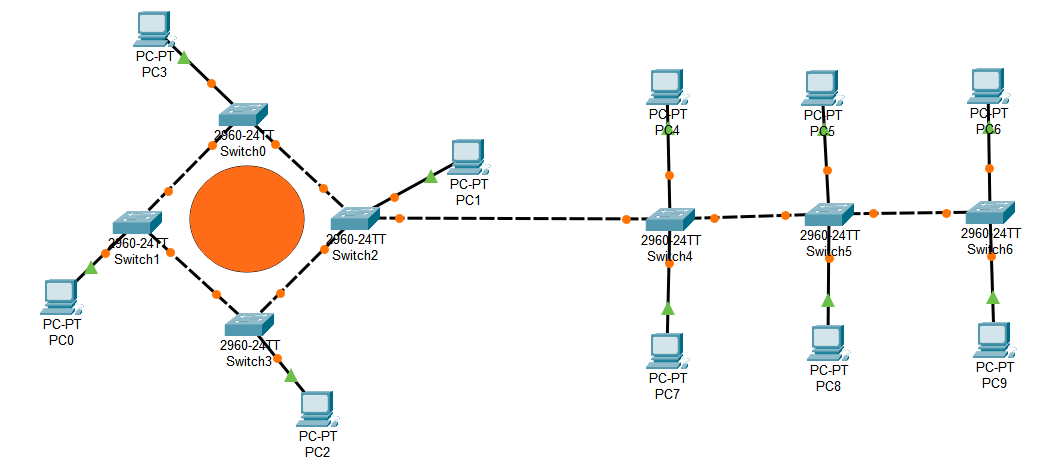
**DIAGRAM:**

****

**PRACTICAL 8**

**AIM**: To perform Hybrid Topology (Bus+Ring)

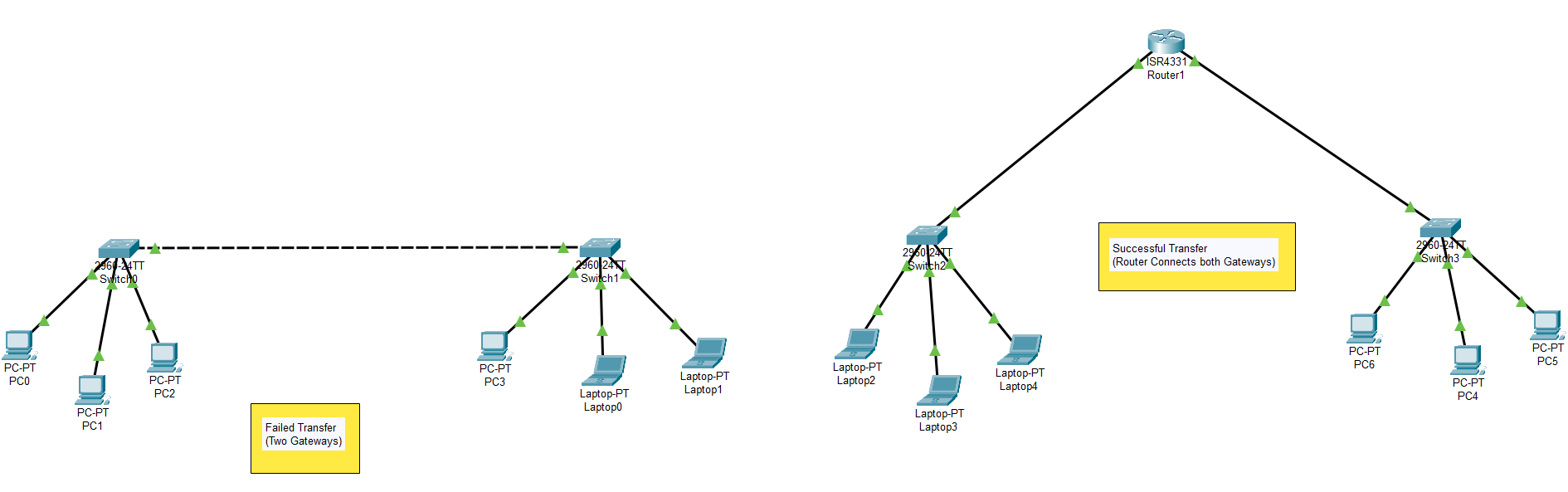
**DIAGRAM:**

****

**PRACTICAL 9**

**AIM**: To perform Message Transfer B/W Two Switches with Router

**DIAGRAM:**

****

**PRACTICAL 10**

**AIM**: To perform python visualization (pythontutor.com)

**CODE:**

n=int(input("enter the no of frames:"))

l=[]

l2=[]

for i in range(0,n):

x=input()

l.append(x)

l2.append(len(x)+1)

sender=''

for i in range(0,n):

sender+=str(l2[i])+l[i]

print(sender)

data=input("enter the message\n")

i=0

inv=0

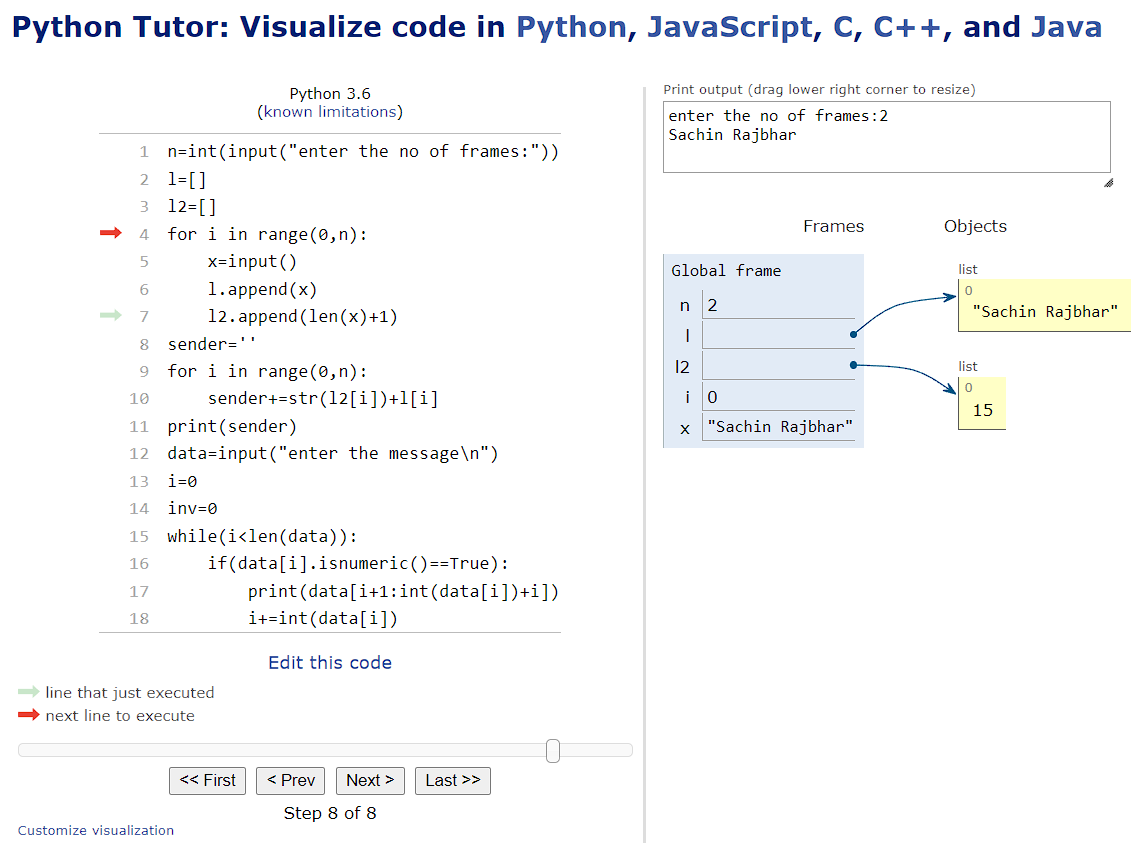
while(i<len(data)):

if(data[i].isnumeric()==True):

print(data[i+1:int(data[i])+i])

i+=int(data[i])

**DIAGRAM:**

****

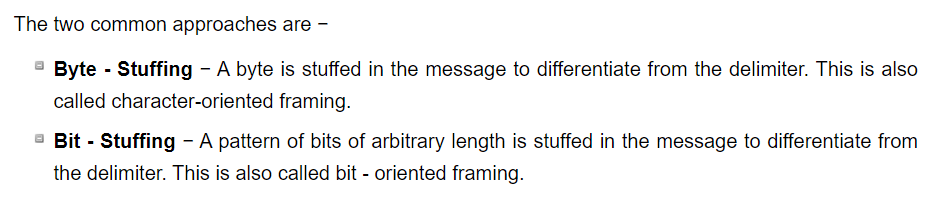
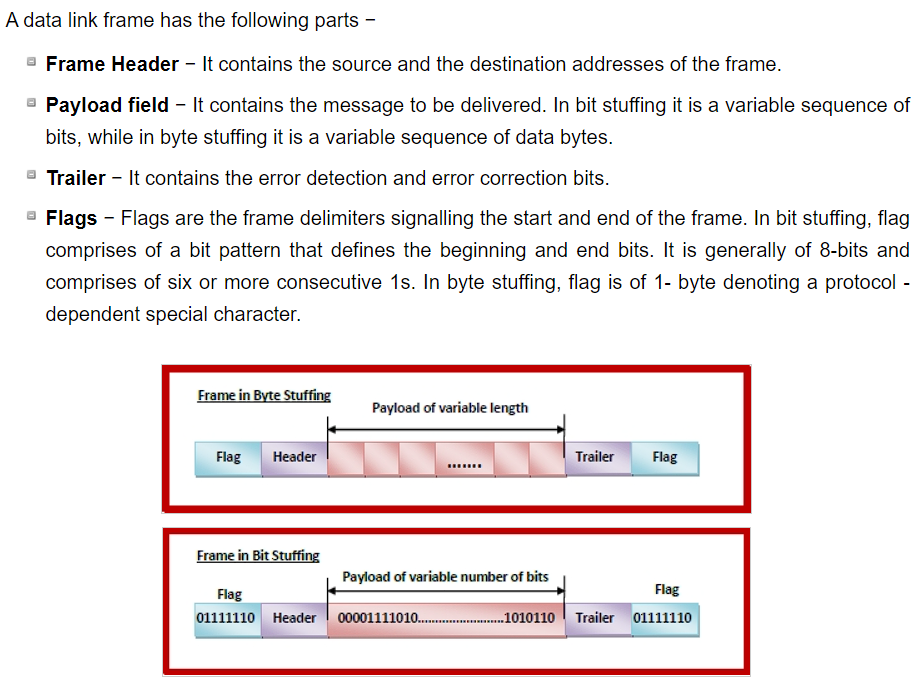
**PRACTICAL 11**

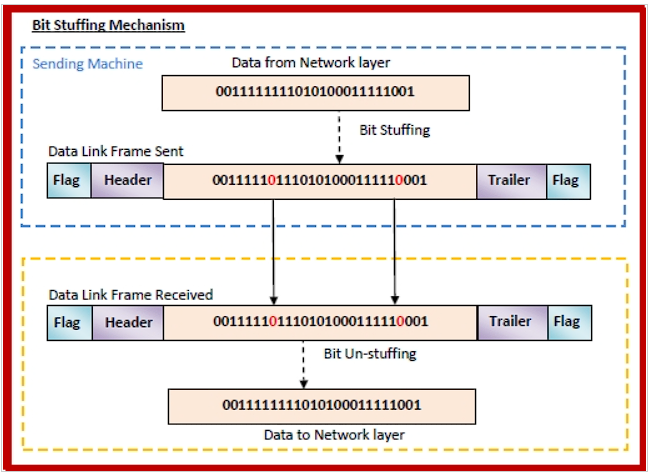
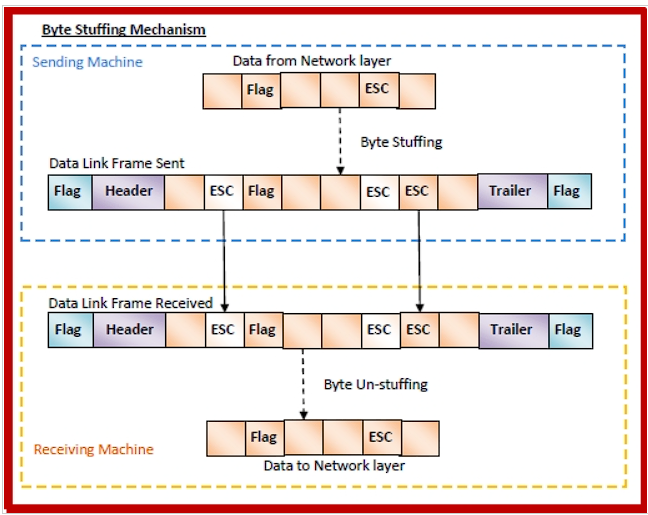
**AIM**: To perform Implementation of Data link framing method - Bit stuffing and Destuffing in python/c/c++.

**DIAGRAM:**

**Byte stuffing** is a mechanism to convert a message formed of a sequence of bytes that may contain reserved values such as frame delimiter, into another byte sequence that does not contain the reserved values.

**Bit stuffing** is the mechanism of inserting one or more non-information bits into a message to be transmitted, to break up the message sequence, for synchronization purpose.



C++ Code for Bit Stuffing:

#include <bits/stdc++.h>

#include <iostream>

using namespace std;

// Function for bit stuffing

void bitStuffing(int N, int arr[])

{

// Stores the stuffed array

int brr[30];

// Variables to traverse arrays

int i, j, k;

i = 0;

j = 0;

// Stores the count of consecutive ones

int count = 1;

// Loop to traverse in the range [0, N)

while (i < N)

{

// If the current bit is a set bit

if (arr[i] == 1)

{

// Insert into array brr[]

brr[j] = arr[i];

// Loop to check for

// next 5 bits

for(k = i + 1; arr[k] == 1 && k < N && count < 5; k++)

{

j++;

brr[j] = arr[k];

count++;

// If 5 consecutive set bits

// are found insert a 0 bit

if (count == 5)

{

j++;

brr[j] = 0;

}

i = k;

}

}

// Otherwise insert arr[i] into

// the array brr[]

else

{

brr[j] = arr[i];

}

i++;

j++;

}

// Print Answer

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

cout << brr[i];

}

// Driver code

int main()

{

int N = 6;

int arr[] = { 1, 1, 1, 1, 1, 1 };

bitStuffing(N, arr);

return 0;

}

C++ Code for Bit Destuffing:

#include <bits/stdc++.h>

using namespace std;

// Function for bit de-stuffing

void bitDestuffing(int N, int arr[])

{

// Stores the de-stuffed array

int brr[30];

// Variables to traverse the arrays

int i, j, k;

i = 0;

j = 0;

// Stores the count of consecutive ones

int count = 1;

// Loop to traverse in the range [0, N)

while (i < N) {

// If the current bit is a set bit

if (arr[i] == 1) {

// Insert into array brr[]

brr[j] = arr[i];

// Loop to check for the next 5 bits

for (k = i + 1; arr[k] == 1 && k < N && count < 5; k++) {

j++;

brr[j] = arr[k];

count++;

// If 5 consecutive set bits are found skip the next bit in arr[]

if (count == 5) {

k++;

}

i = k;

}

}

// Otherwise insert arr[i] into the array brr

else {

brr[j] = arr[i];

}

i++;

j++;

}

// Print Answer

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

cout<< brr[i];

}

// Driver Code

int main()

{

int N = 7;

int arr[] = { 1, 1, 1, 1, 1, 0, 1 };

bitDestuffing(N, arr);

return 0;

}

**PRACTICAL 12**

**AIM**: Implementation of Error detection method - even and odd parity in python.

Code:

# Parity is no. of 1's

# Function to check odd parity

# It returns 1 if n has odd parity,else returns 0

def getParity( n ):

    parity = 0

    while n:

        parity = ~parity

        n = n & (n - 1)

    return parity

# Driver program

n = int(input("Enter a number: "))

print ("Parity of ", n," = ",

    ( "odd" if getParity(n) else "even"))

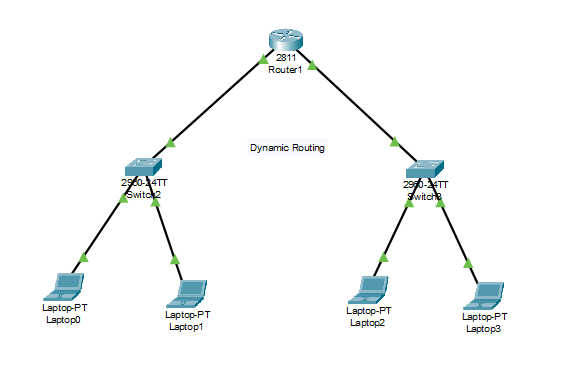
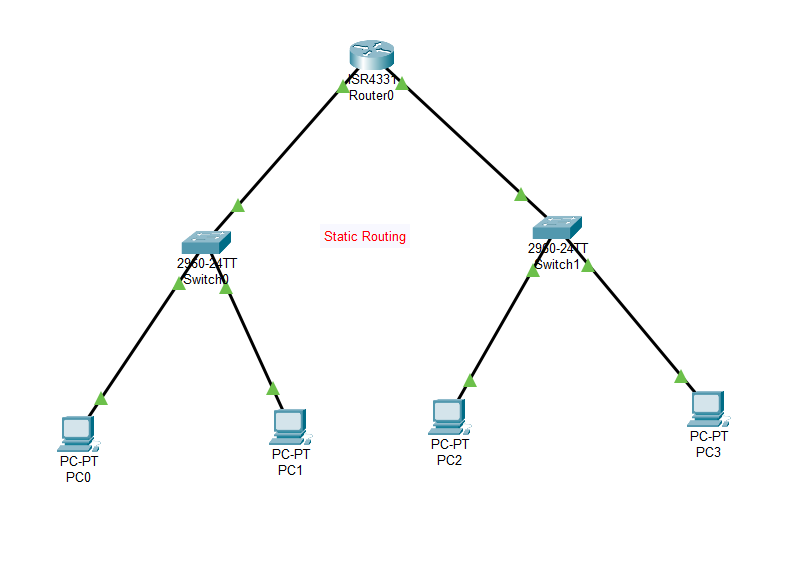
Output:



**PRACTICAL 13**

**AIM**: To perform Static and Dynamic Routing

**DIAGRAM:**

****

**PRACTICAL 9**

**AIM**: To perform Message Transfer B/W Two Switches with Router

**DIAGRAM:**

**PRACTICAL 9**

**AIM**: To perform Message Transfer B/W Two Switches with Router

**DIAGRAM:**

**PRACTICAL 9**

**AIM**: To perform Message Transfer B/W Two Switches with Router

**DIAGRAM:**

**PRACTICAL 9**

**AIM**: To perform Message Transfer B/W Two Switches with Router

**DIAGRAM:**

**PRACTICAL 9**

**AIM**: To perform Message Transfer B/W Two Switches with Router

**DIAGRAM:**

**PRACTICAL 9**

**AIM**: To perform Message Transfer B/W Two Switches with Router

**DIAGRAM:**

**PRACTICAL 9**

**AIM**: To perform Message Transfer B/W Two Switches with Router

**DIAGRAM:**