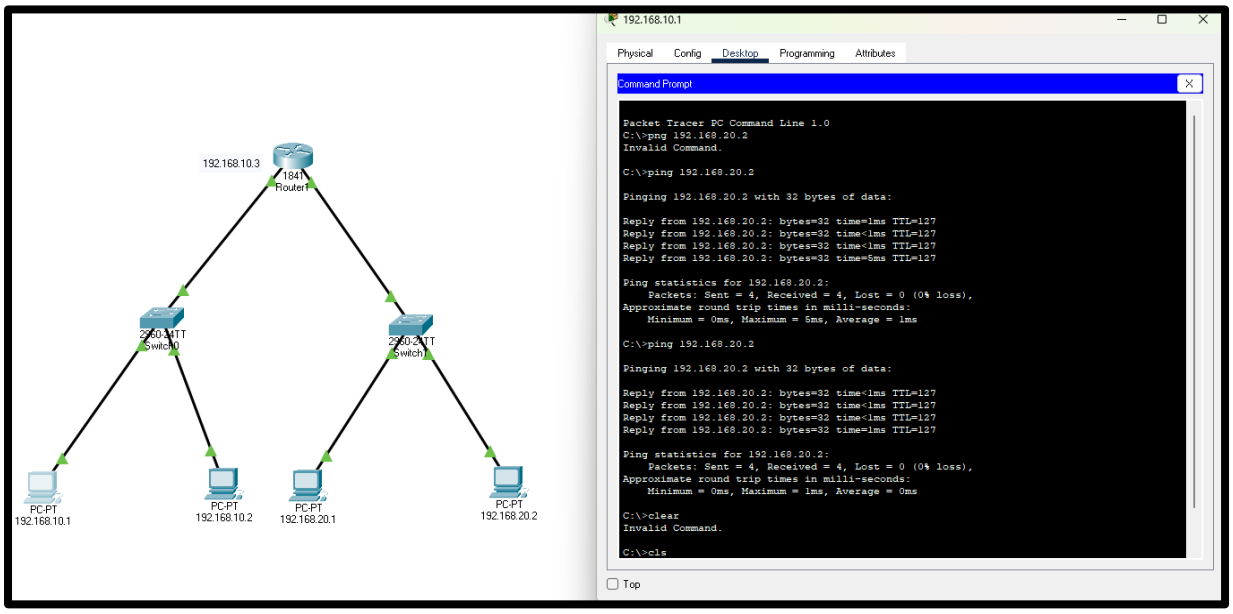
Date: 9/12/2025

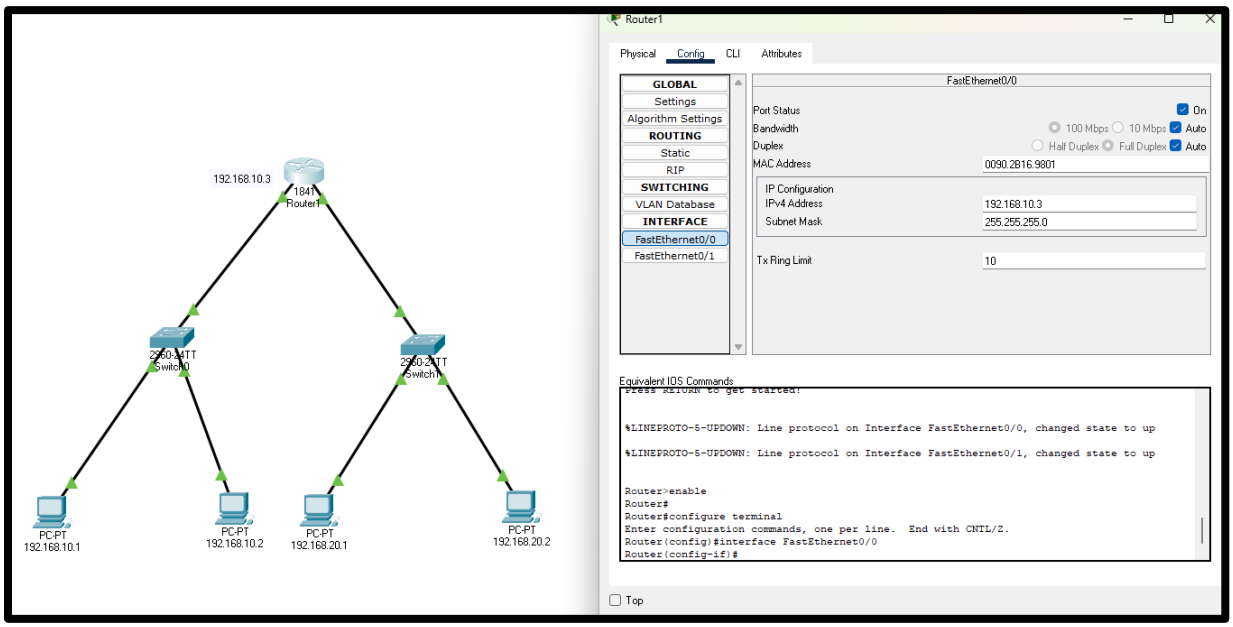
Lab Practical #11:

Study the concept of routing using packet tracer. (Dynamic Routing)

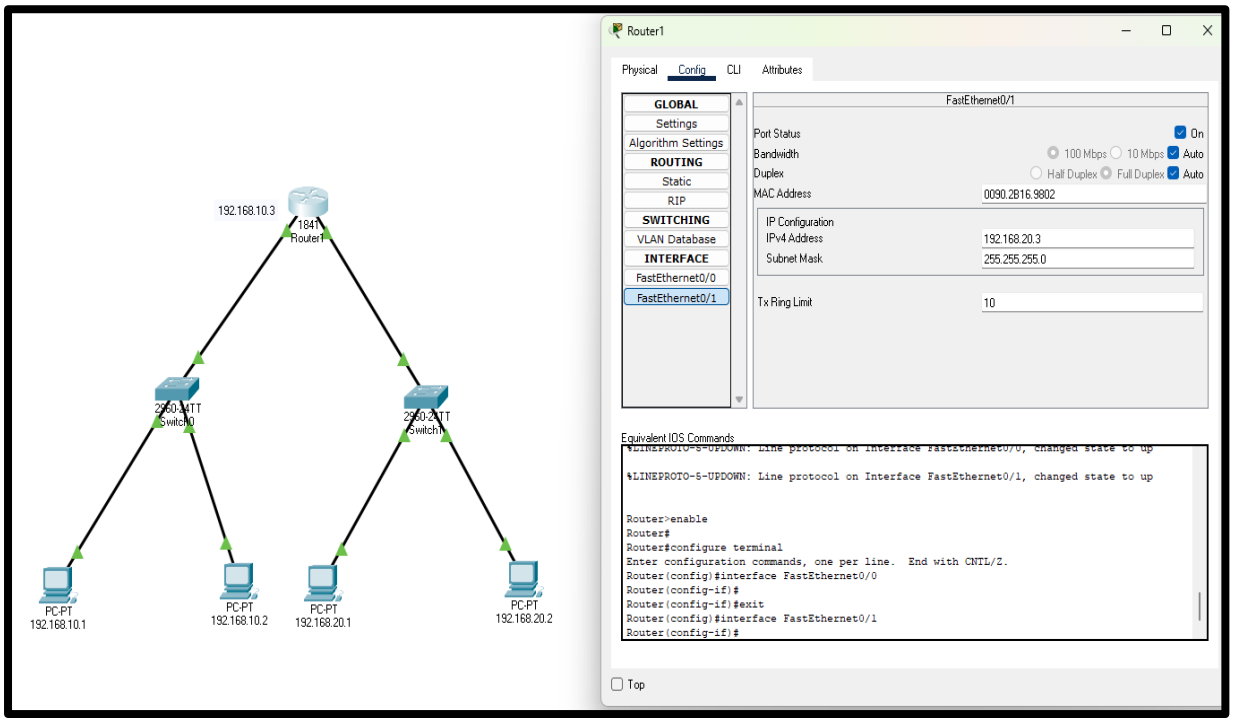
Practical Assignment #11:

1. **Connect the two different networks based on the calculated IP addresses and subnet using a packet tracer.**

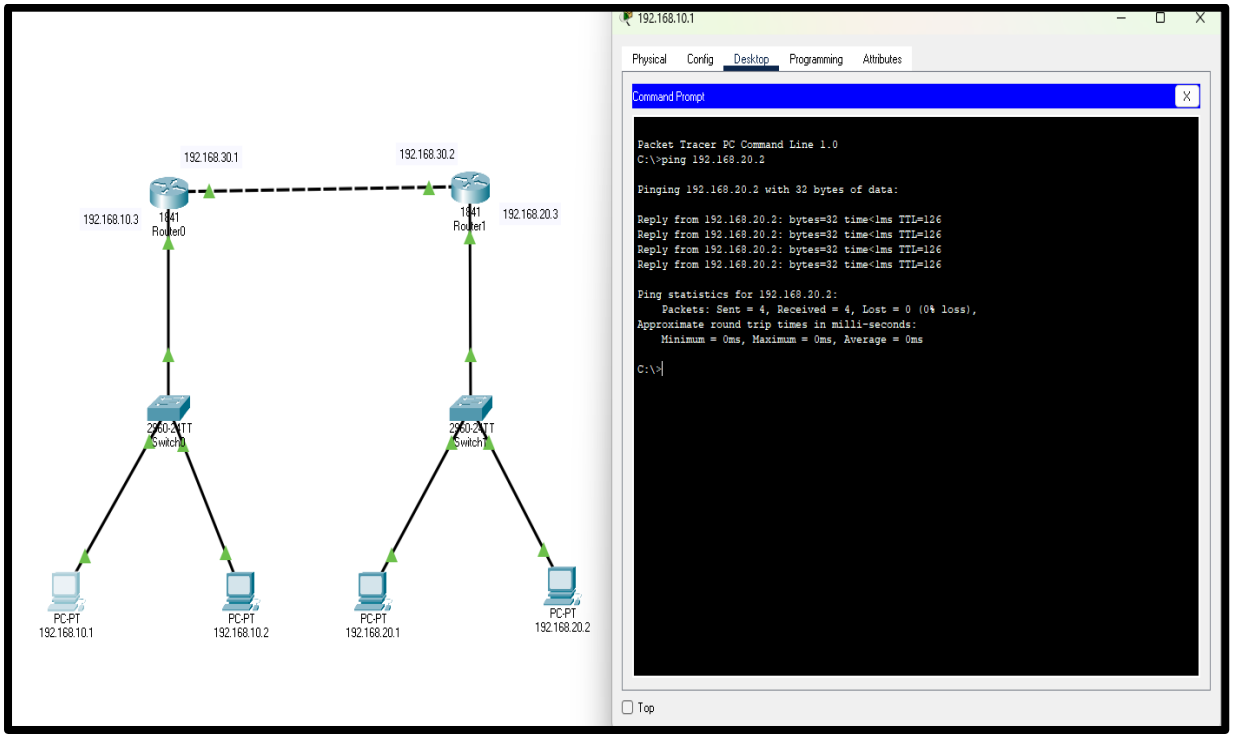




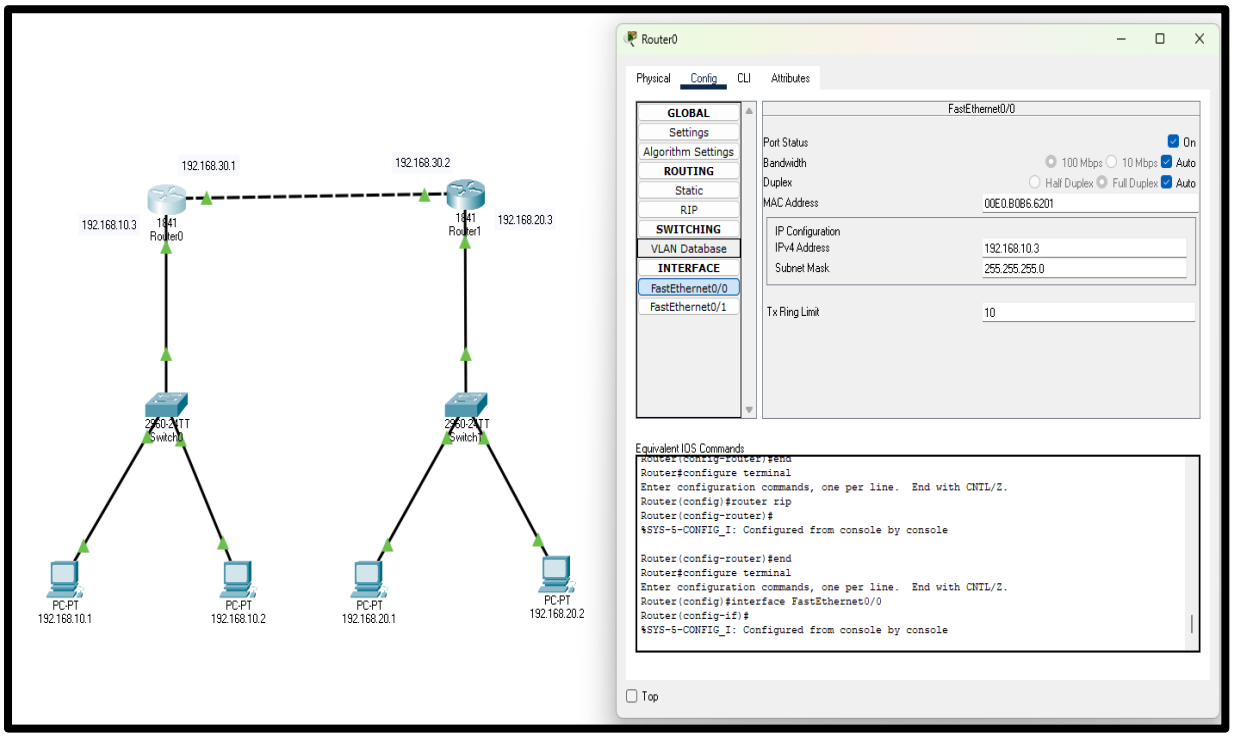
Date: 9/12/2025

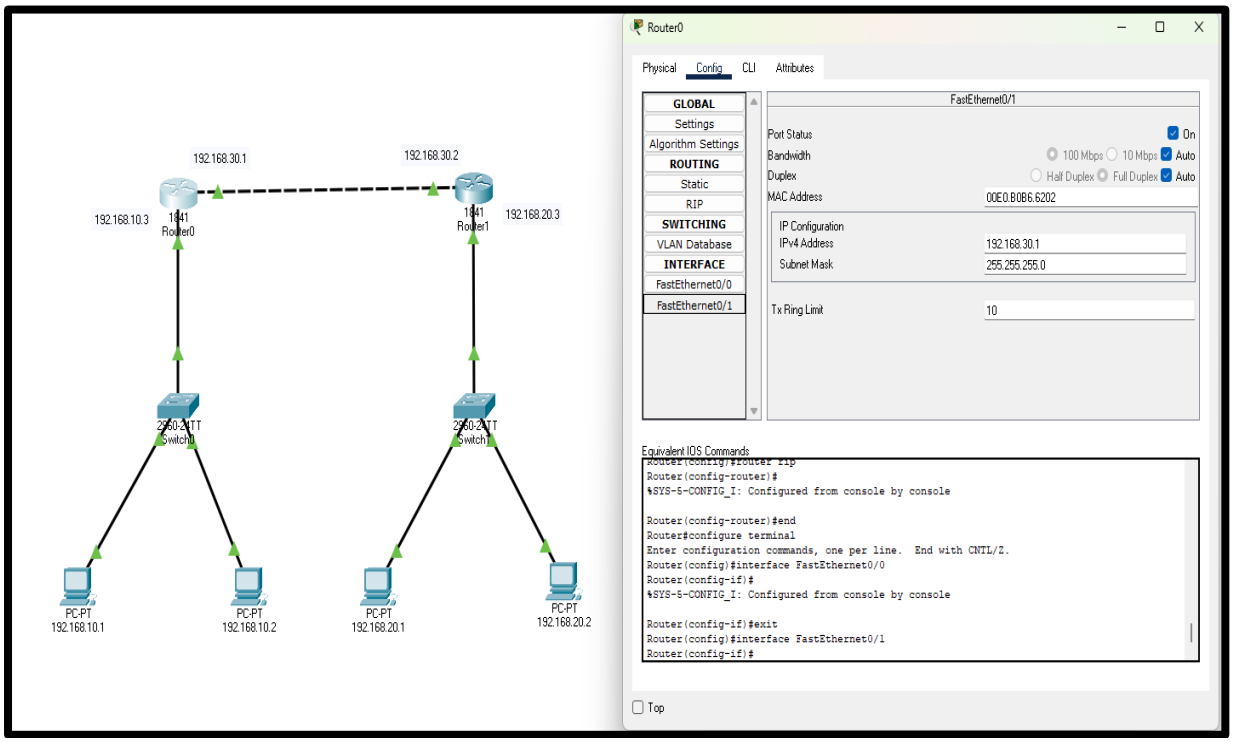


2. Connect the three different networks based on the calculated IP addresses and subnet using a packet tracer.

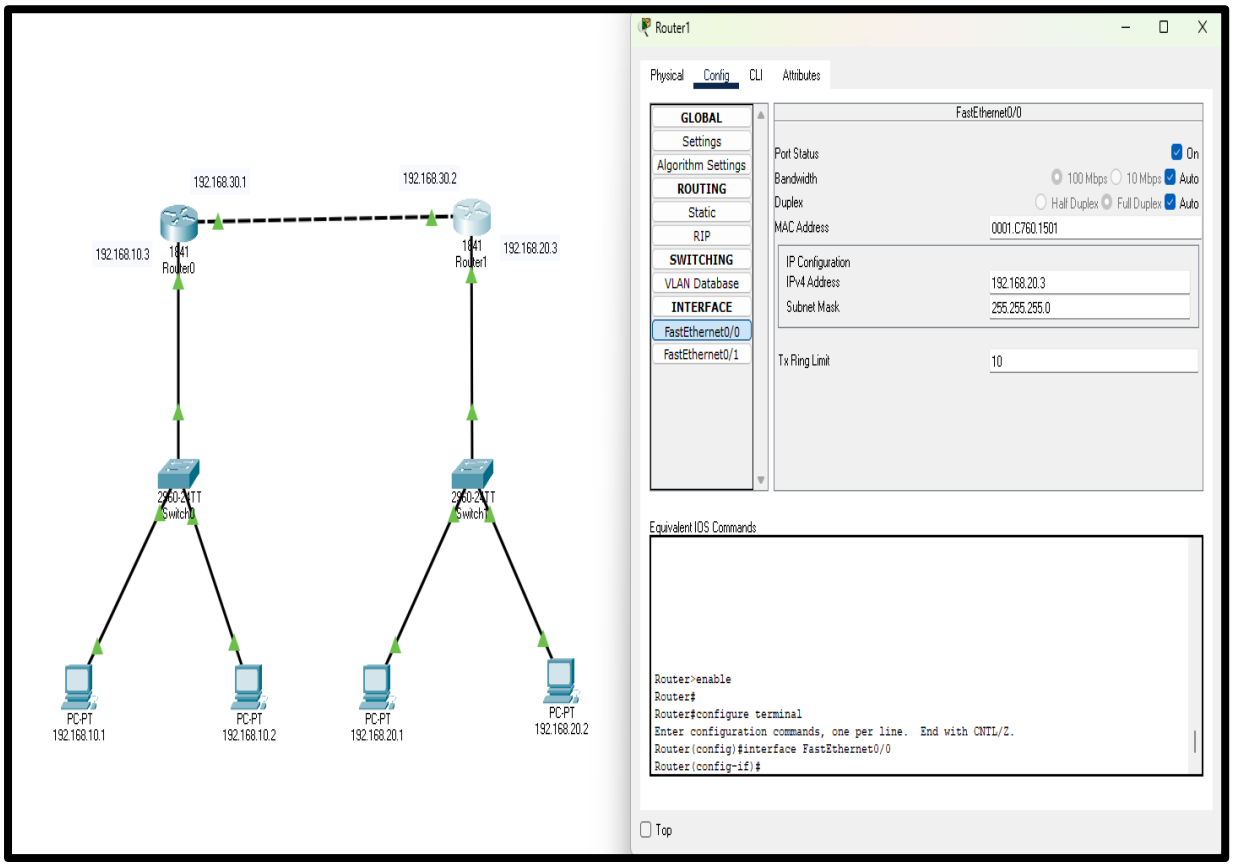


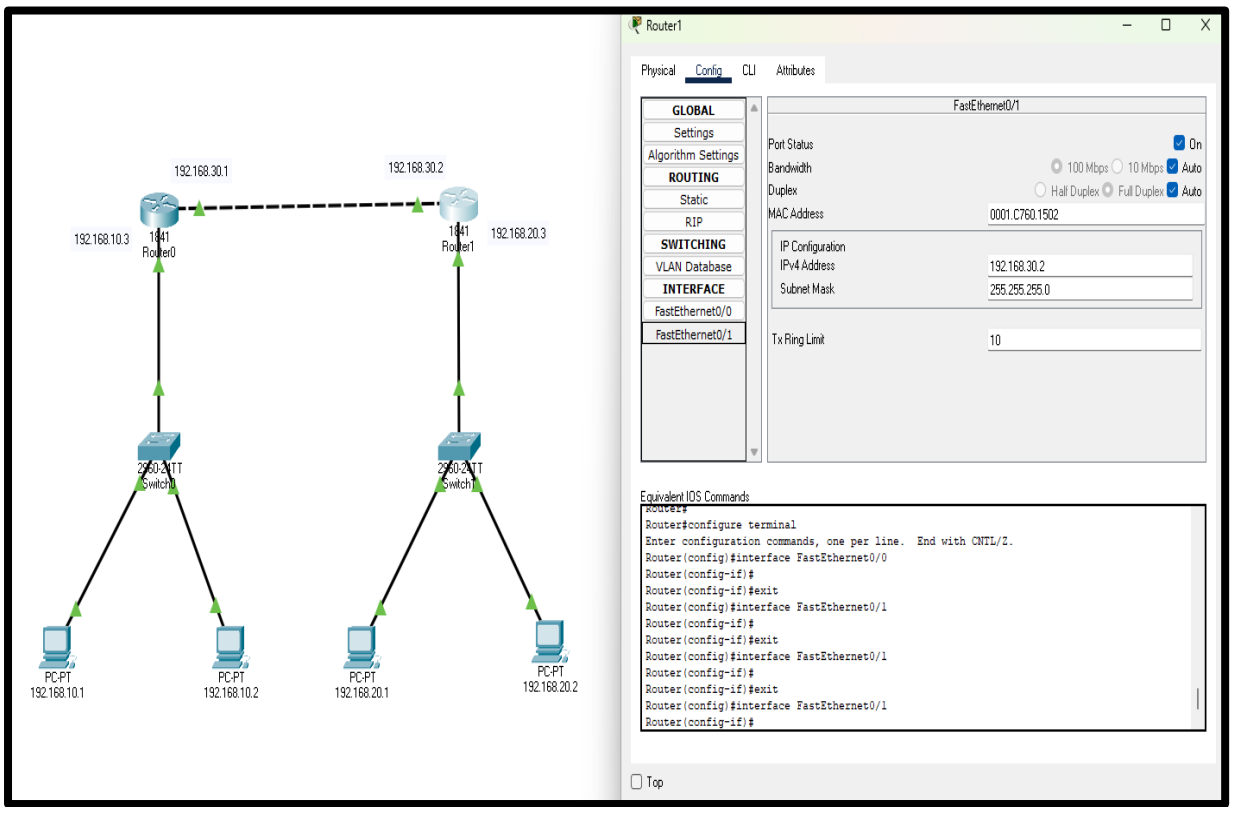
Date: 9/12/2025



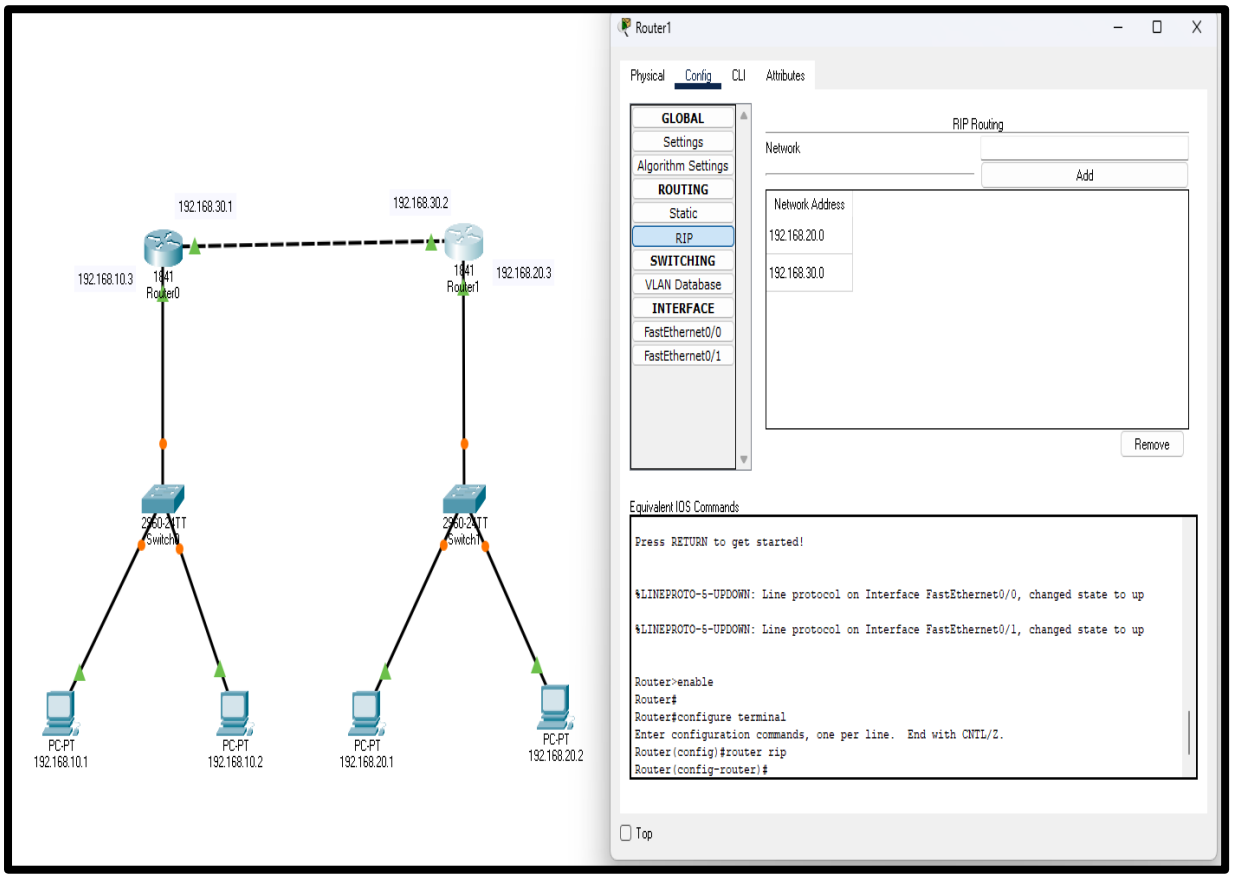


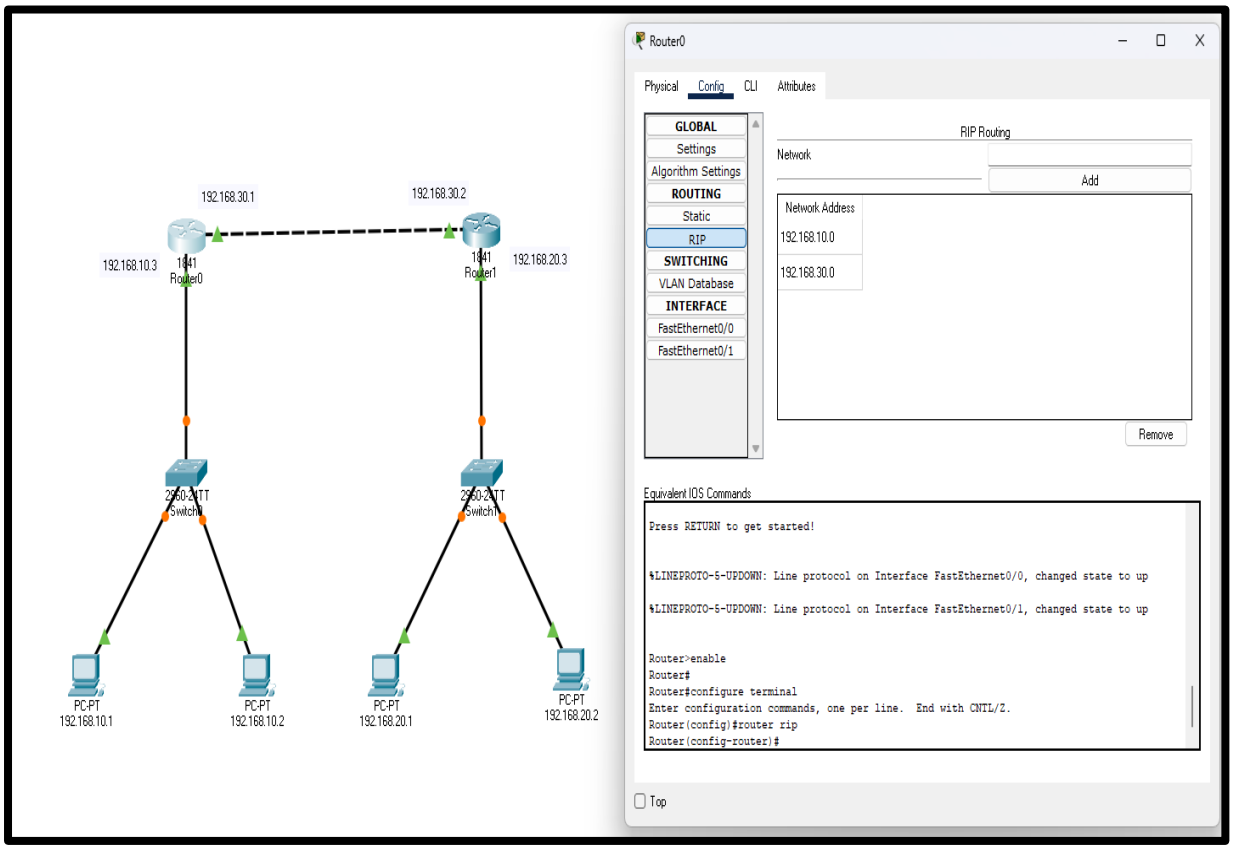
Date: 9/12/2025





Date: 9/12/2025





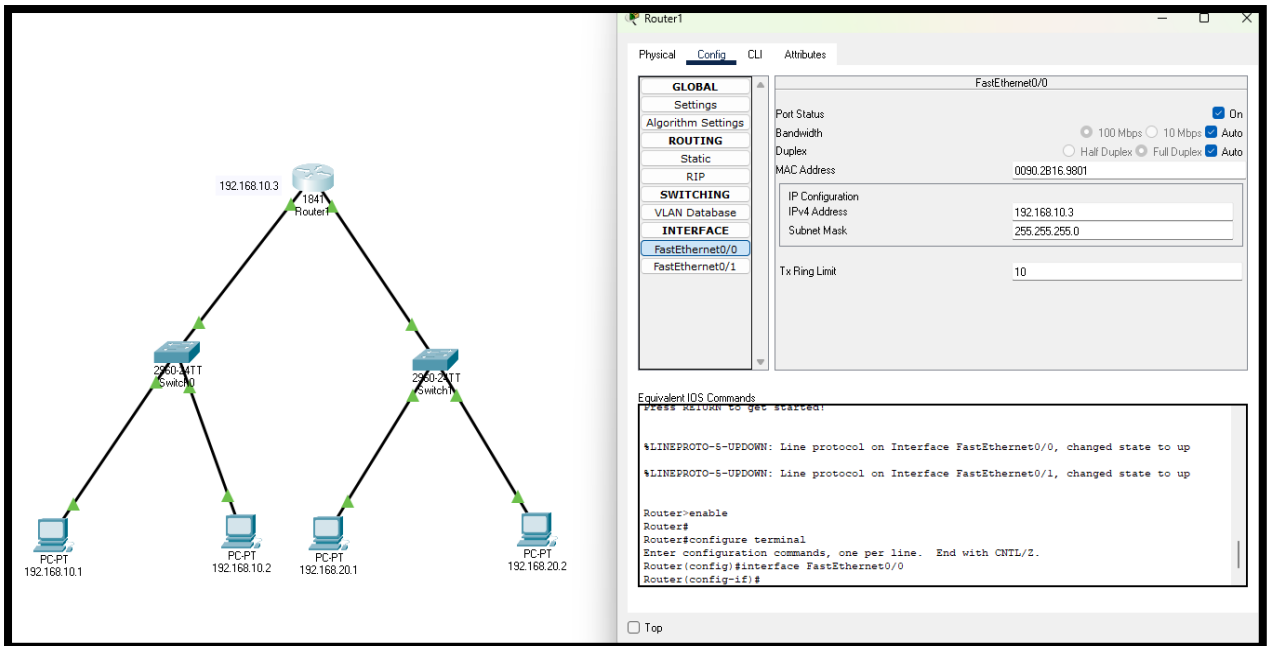
Date: 9/15/2025

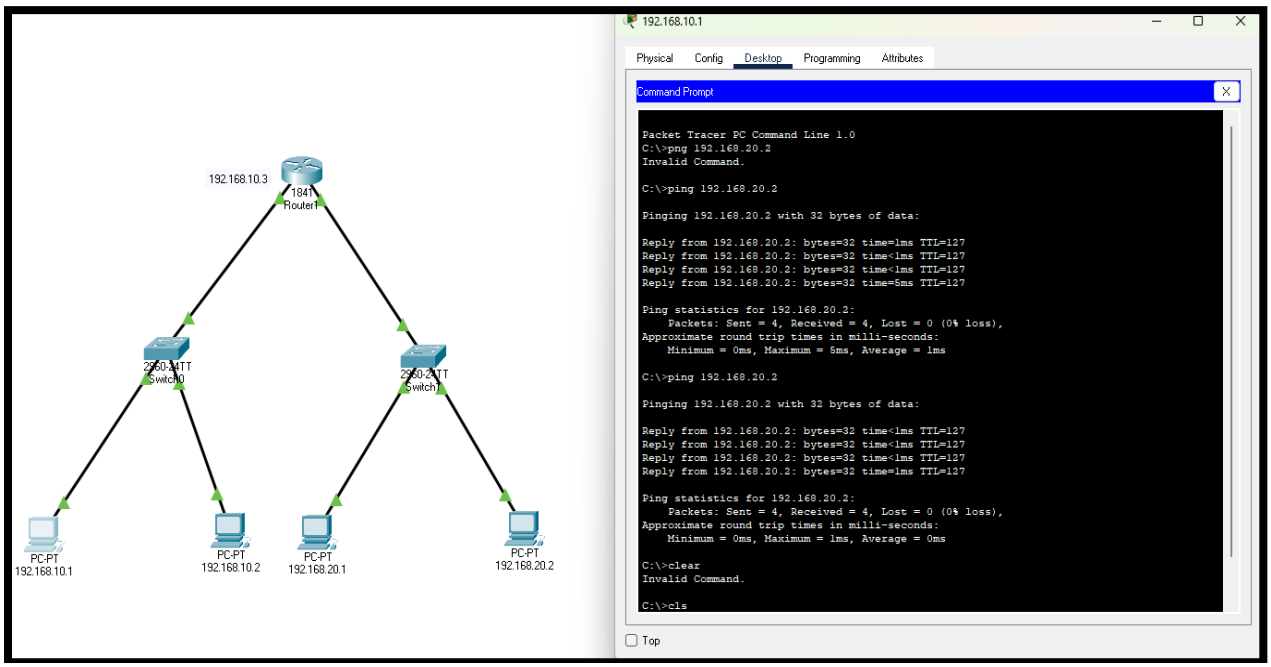
Lab Practical #12:

Study the concept of routing using packet tracer. (Static Routing)

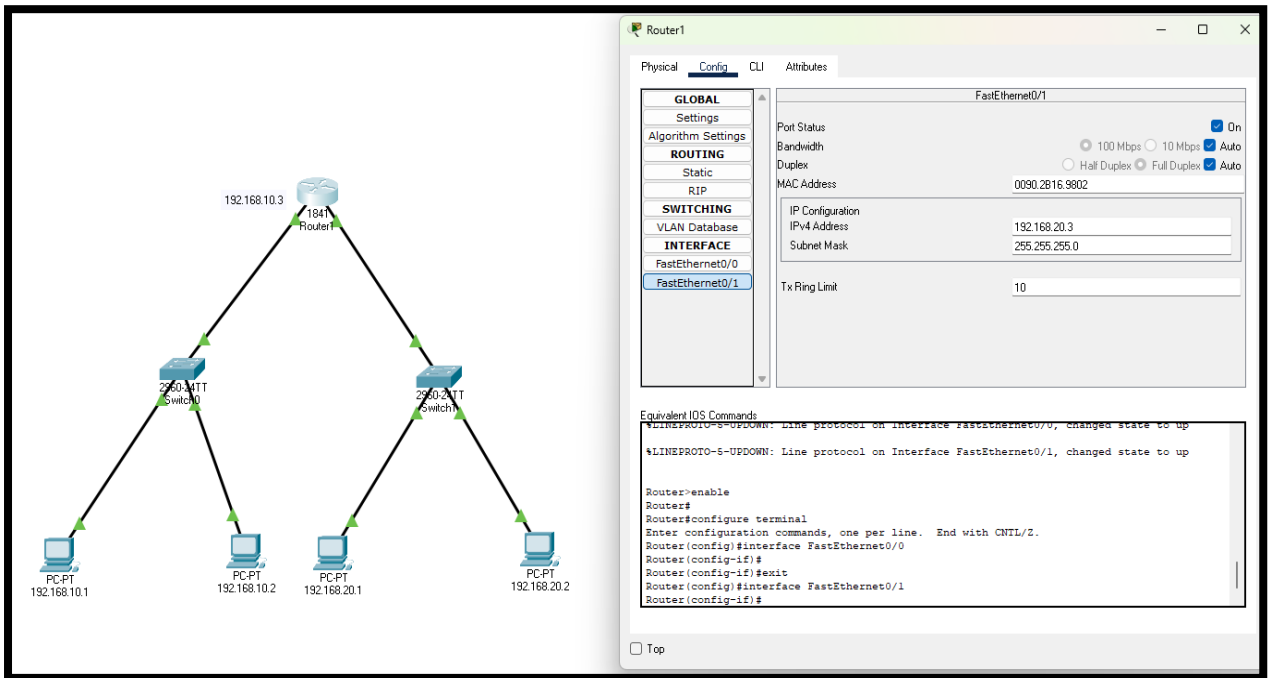
Practical Assignment #12:

1. **Connect the two different networks based on the calculated IP addresses and subnet using a packet tracer.**

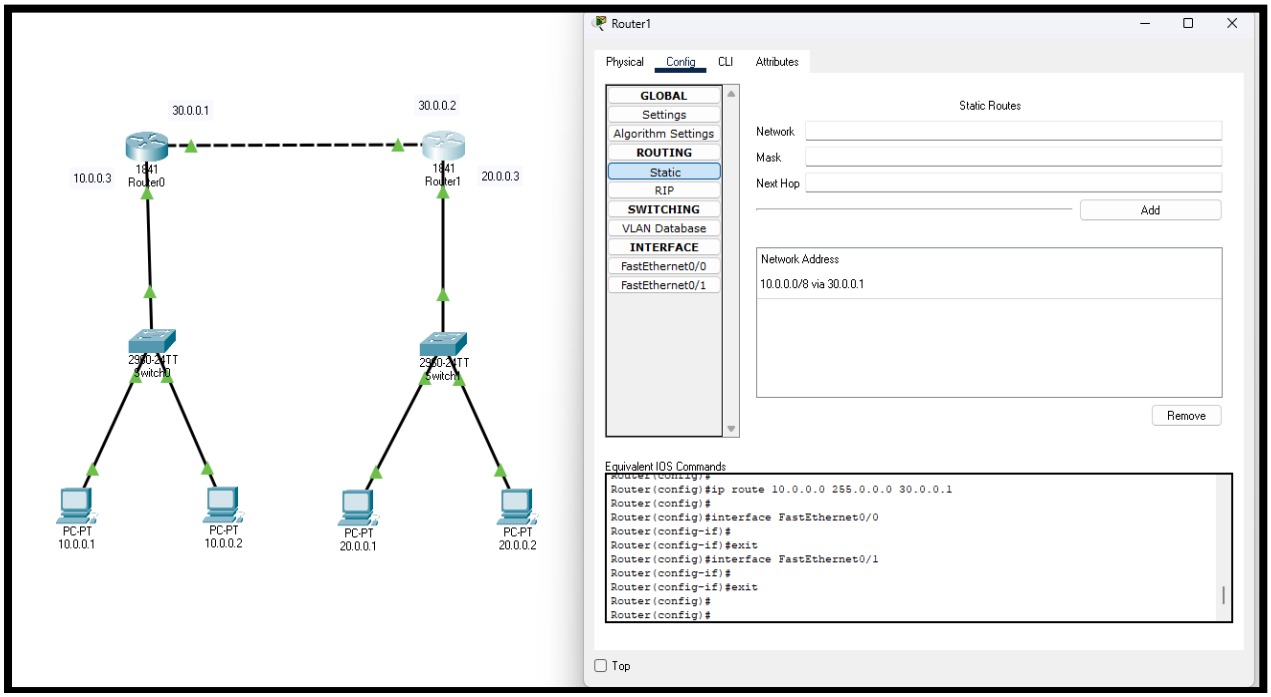




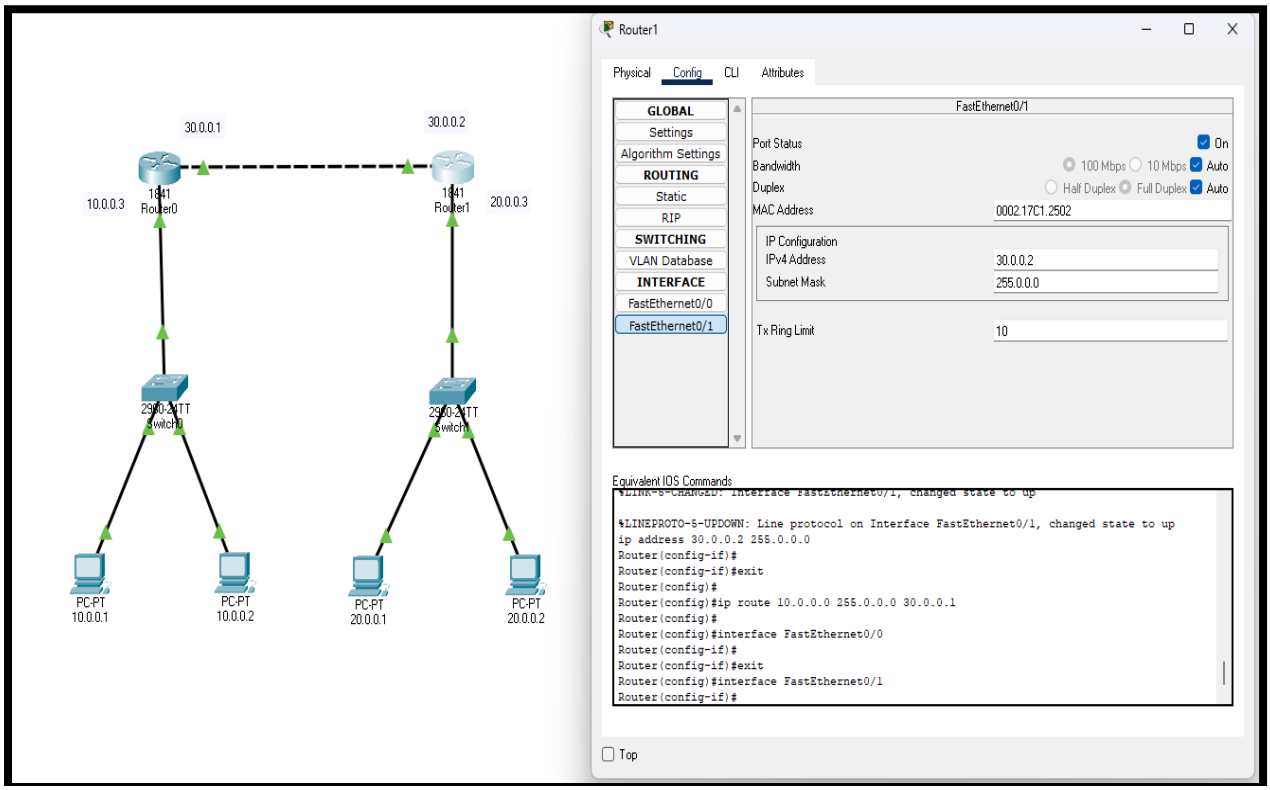
Date: 9/15/2025

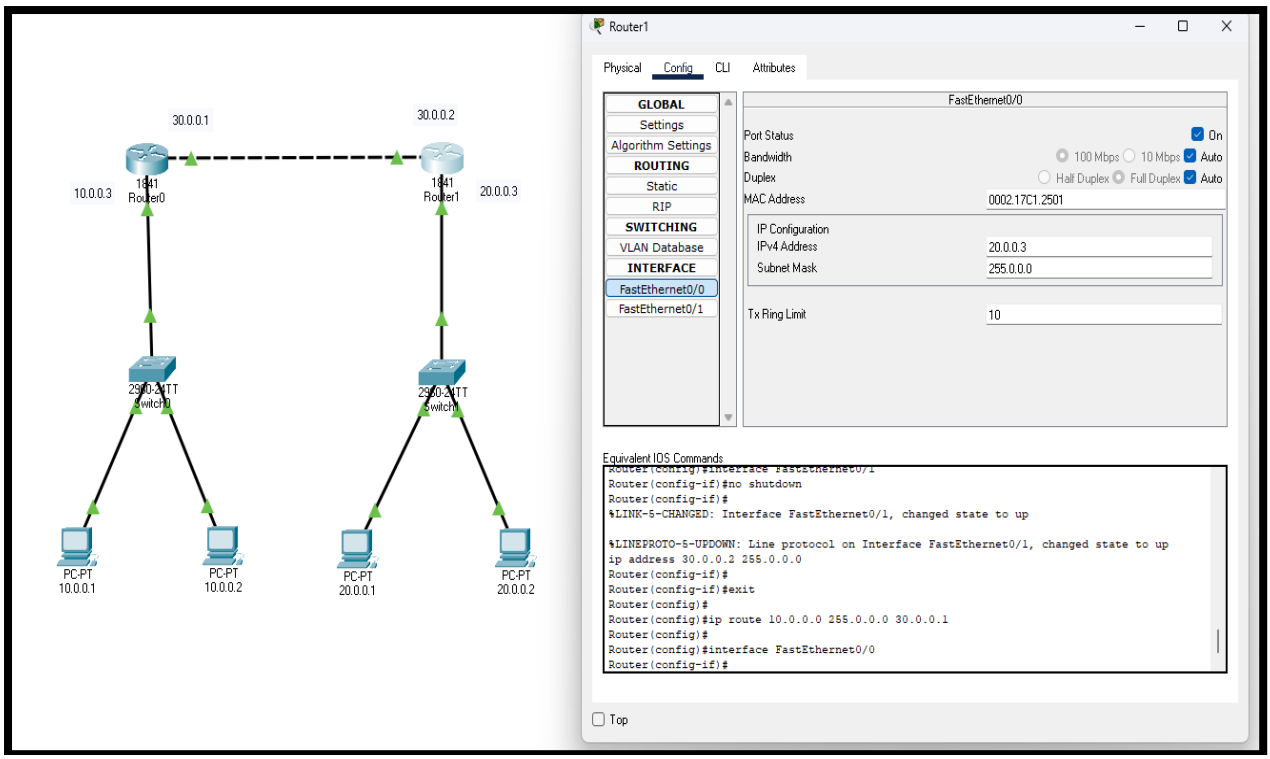


2. **Connect the three different networks based on the calculated IP addresses and subnet using a packet tracer.**

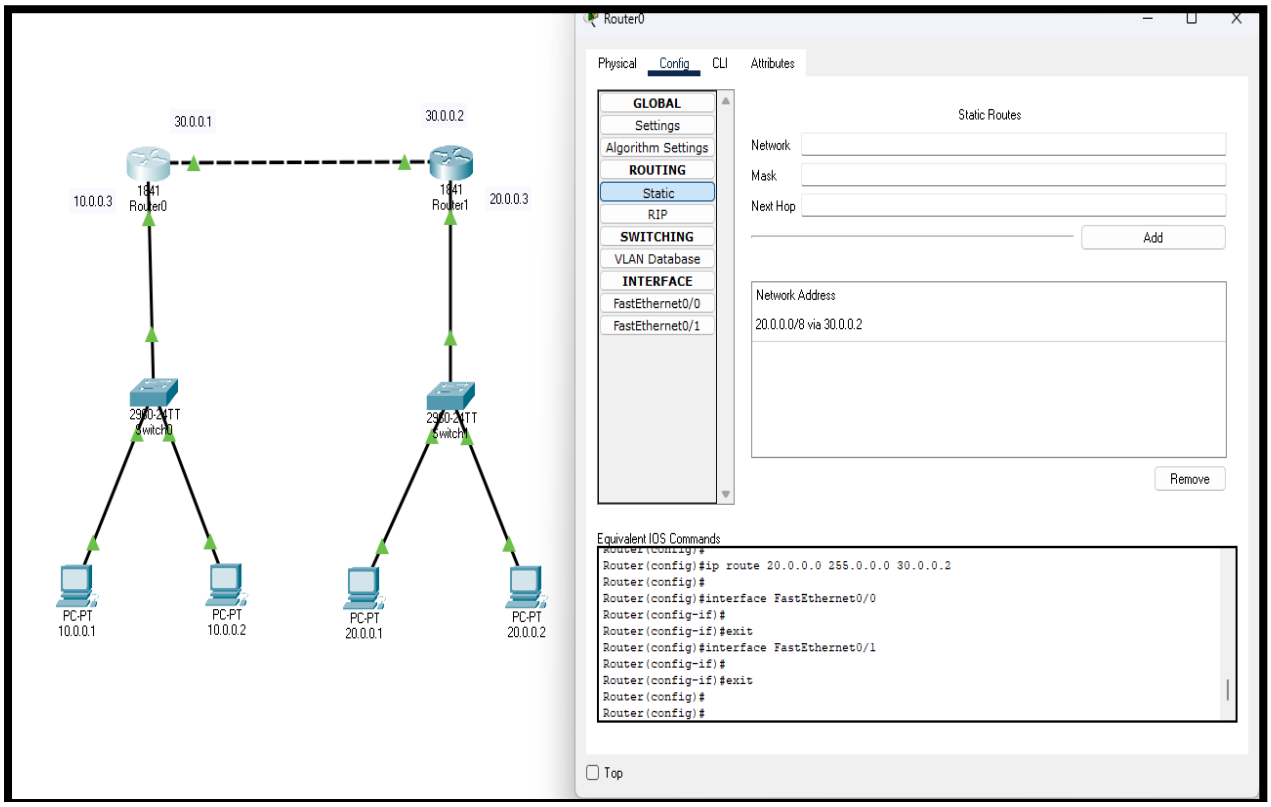


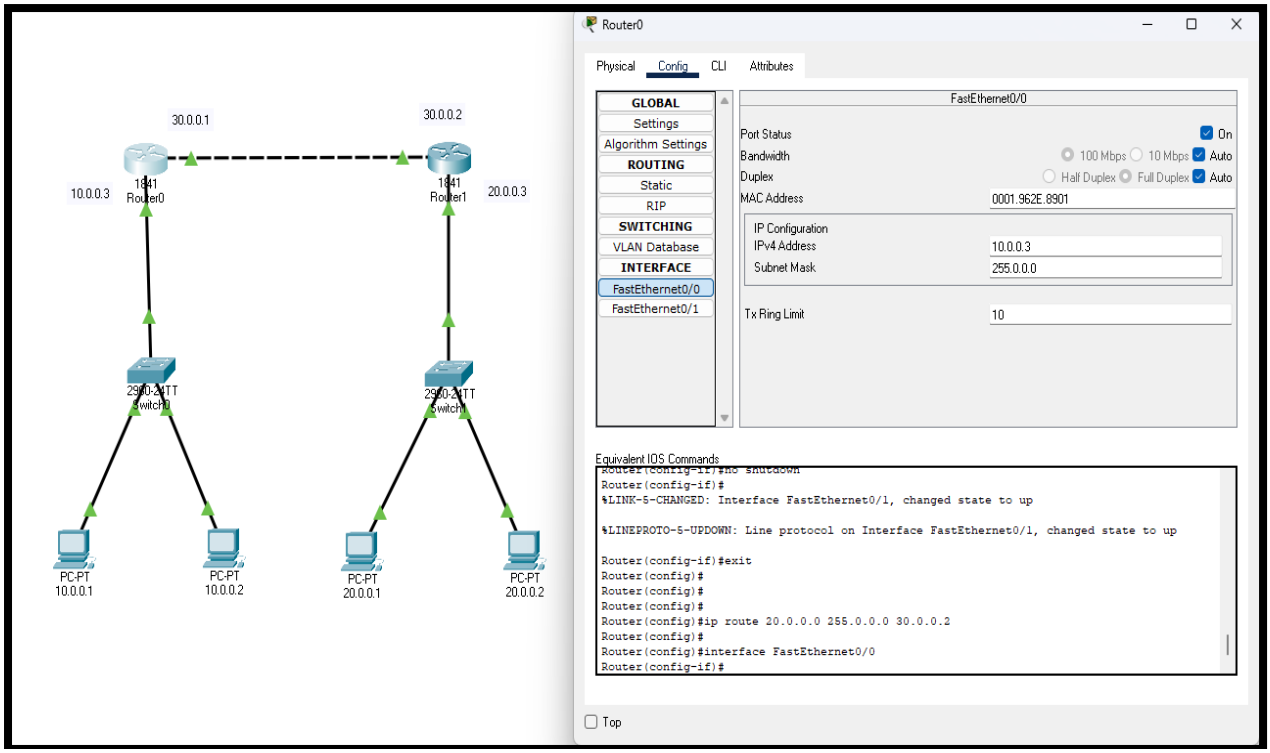
Date: 9/15/2025



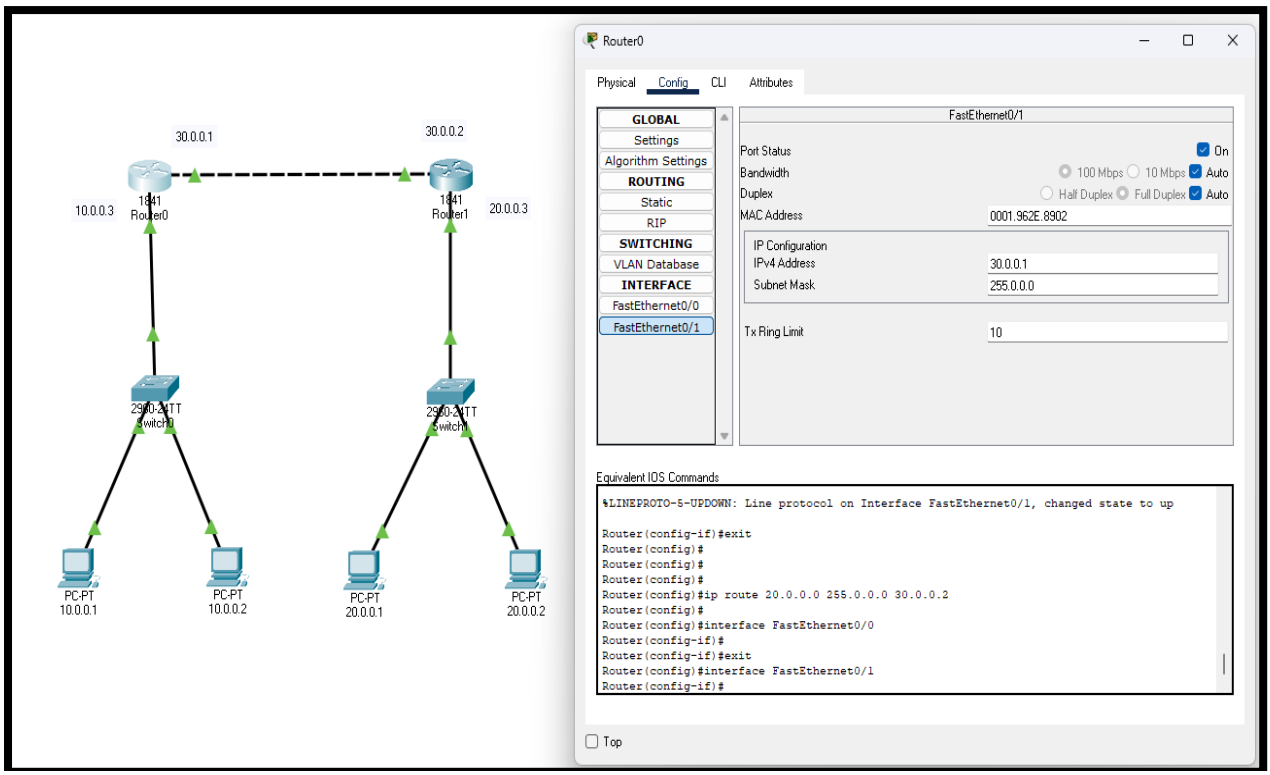


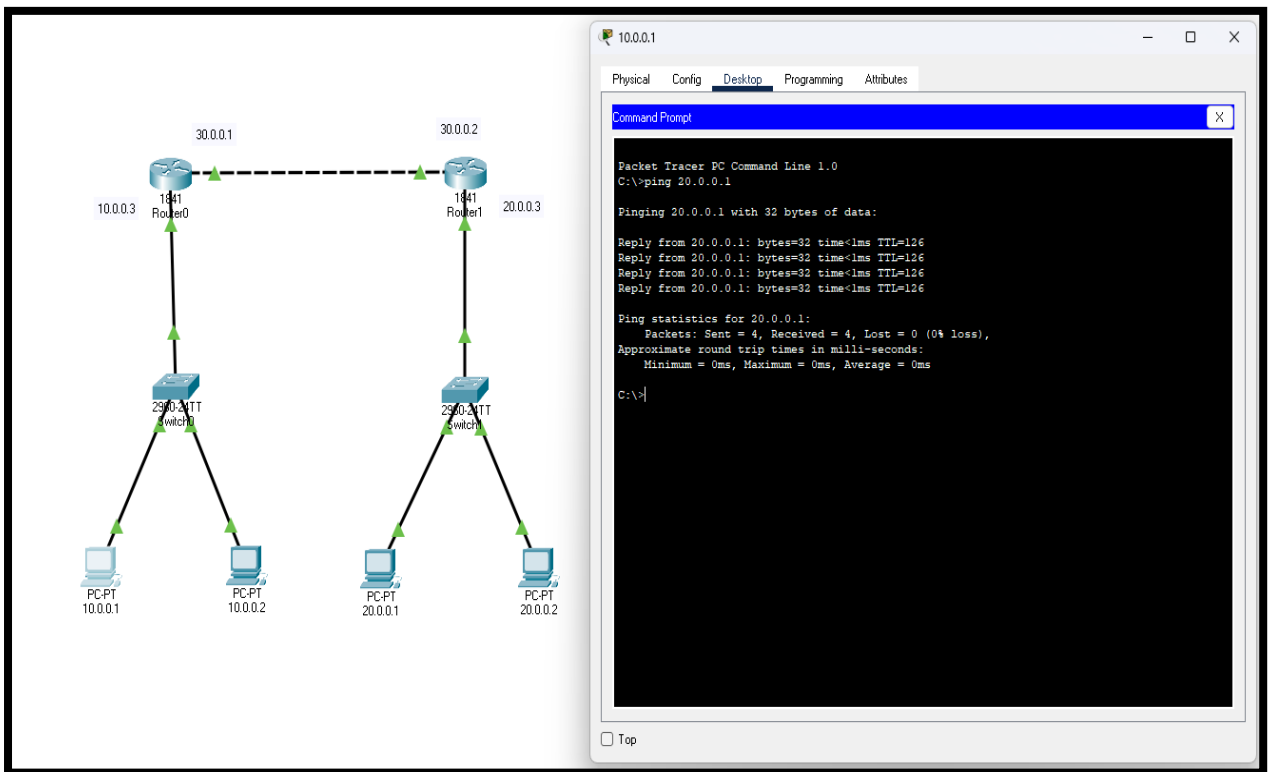
Date: 9/15/2025





Date: 9/15/2025





Date: 17/09/2025

Lab Practical #13:

To develop network using distance vector routing protocol and link state routing protocol.

Practical Assignment #13:

1. **C/Java Program: Distance Vector Routing Algorithm using Bellman Ford's Algorithm.**

```
import java.util.*;

public class DistanceVectorRouting {
   private static final int INF = 9999;

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the number of routers: ");
      int numRouters = scanner.nextInt();

      int[][] costMatrix = new int[numRouters][numRouters];
      System.out.println("Enter the cost matrix (use " + INF + " for infinity):");
      for (int i = 0; i < numRouters; i++) {
         for (int j = 0; j < numRouters; j++) {
            costMatrix[i][j] = scanner.nextInt();
         }
      }

      int[][] distanceVector = new int[numRouters][numRouters];
      int[][] nextHop = new int[numRouters][numRouters];

      for (int i = 0; i < numRouters; i++) {
         for (int j = 0; j < numRouters; j++) {
            distanceVector[i][j] = costMatrix[i][j];
            nextHop[i][j] = (costMatrix[i][j] != INF && i != j) ? j : -1;
         }
      }

      boolean updated;
      do {
         updated = false;
         for (int i = 0; i < numRouters; i++) {
```

Date: 17/09/2025

```
            for (int j = 0; j < numRouters; j++) {
                for (int k = 0; k < numRouters; k++) {
                    if (distanceVector[i][k] + distanceVector[k][j] <
distanceVector[i][j]) {
                        distanceVector[i][j] = distanceVector[i][k] +
distanceVector[k][j];
                        nextHop[i][j] = nextHop[i][k];
                        updated = true;
                    }
                }
            }
        }
    } while (updated);

    System.out.println("\nFinal Distance Vector Table:");
    for (int i = 0; i < numRouters; i++) {
        System.out.println("Router " + (i + 1) + ":");
        for (int j = 0; j < numRouters; j++) {
            if (distanceVector[i][j] == INF) {
                System.out.print("INF ");
            } else {
                System.out.print((distanceVector[i][j] + 1) + " ");
            }
        }
        System.out.println();
    }

    scanner.close();
  }
}
```

Date: 17/09/2025

Output:

2. C/Java Program: Link state routing algorithm.

```
import java.util.*;

public class LinkStateRouting {
   private static final int INF = 9999;

   private static int minDistance(int[] dist, boolean[] visited, int n) {
      int min = INF, minIndex = -1;
      for (int v = 0; v < n; v++) {
         if (!visited[v] && dist[v] < min) {
            min = dist[v];
            minIndex = v;
         }
      }
      return minIndex;
   }

   private static void printPath(int[] prev, int j) {
      if (prev[j] == -1) {
         System.out.print((j + 1));
         return;
      }
```

```
        printPath(prev, prev[j]);
        System.out.print(" -> " + (j + 1));
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of routers: ");
        int numRouters = scanner.nextInt();

        int[][] costMatrix = new int[numRouters][numRouters];
        System.out.println("Enter the cost matrix (use " + INF + " for infinity):");
        for (int i = 0; i < numRouters; i++) {
            for (int j = 0; j < numRouters; j++) {
                costMatrix[i][j] = scanner.nextInt();
            }
        }

        for (int src = 0; src < numRouters; src++) {
            int[] dist = new int[numRouters];
            int[] prev = new int[numRouters];
            boolean[] visited = new boolean[numRouters];

            Arrays.fill(dist, INF);
            Arrays.fill(prev, -1);

            dist[src] = 0;

            for (int count = 0; count < numRouters - 1; count++) {
                int u = minDistance(dist, visited, numRouters);
                if (u == -1) break;
                visited[u] = true;

                for (int v = 0; v < numRouters; v++) {
                    if (!visited[v] && costMatrix[u][v] != INF && dist[u] != INF &&
                        dist[u] + costMatrix[u][v] < dist[v]) {
                        dist[v] = dist[u] + costMatrix[u][v];
                        prev[v] = u;
                    }
                }
```

Date: 17/09/2025

```
            }

            System.out.println("\nRouting Table for Router " + (src + 1) + ":");
            System.out.println("Destination\tCost\tPath");
            for (int dest = 0; dest < numRouters; dest++) {
                if (dist[dest] == INF) {
                    System.out.println((dest + 1) + "\t\tINF\tNo path");
                } else {
                    System.out.print((dest + 1) + "\t\t" + dist[dest] + "\t");
                    printPath(prev, dest);
                    System.out.println();
                }
            }
        }

        scanner.close();
    }
}
```

Output:

```
D:\CN\LinkState.exe
 Enter the cost matrix values:
0->0:1

0->1:5

0->2:4

1->0:6

1->1:7

1->2:4

2->0:3

2->1:2

2->2:6

 Enter the source router:4

4==>0:Path taken:0
<--4
 Shortest path cost:0
4==>1:Path taken:1
<--4
 Shortest path cost:0
4==>2:Path taken:2
<--4
 Shortest path cost:0
--------------------------------
Process exited after 123.1 seconds with return value 3
Press any key to continue . . .
```

Date: 18/09/2025

Lab Practical #14:

Implementation of parity bit check Using C/Java language with example.

Practical Assignment #14:

C/Java Program: Implementation of parity bit check Using C/Java language.

```
import java.util.Scanner;

public class ParityBitCheck {

    public static String addParityBit(String dataBits, String parityType) {
        int countOnes = 0;
        for (char bit : dataBits.toCharArray()) {
            if (bit == '1') countOnes++;
        }

        char parityBit;

        if (parityType.equalsIgnoreCase("even")) {
            parityBit = (countOnes % 2 == 0) ? '0' : '1';
        } else {
            parityBit = (countOnes % 2 == 0) ? '1' : '0';
        }

        return dataBits + parityBit;
    }
    public static boolean checkParity(String data, String parityType) {
        char parityBit = data.charAt(data.length() - 1);
        String dataBits = data.substring(0, data.length() - 1);

        int countOnes = 0;
```

```
        for (char bit : dataBits.toCharArray()) {
            if (bit == '1') countOnes++;
        }

        boolean isValid;
        if (parityType.equalsIgnoreCase("even")) {
            isValid = ((countOnes % 2 == 0) && parityBit == '0') ||
                    ((countOnes % 2 == 1) && parityBit == '1');
        } else {
            isValid = ((countOnes % 2 == 0) && parityBit == '1') ||
                    ((countOnes % 2 == 1) && parityBit == '0');
        }

        return isValid;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter binary data (without parity bit): ");
        String dataBits = scanner.nextLine();

        System.out.print("Enter parity type (even/odd): ");
        String parityType = scanner.nextLine();

        String dataWithParity = addParityBit(dataBits, parityType);
        System.out.println("Data with " + parityType + " parity bit: " +
dataWithParity);

        if (checkParity(dataWithParity, parityType)) {
            System.out.println("Parity check PASSED.");
```

Date: 18/09/2025

```
        } else {
            System.out.println("Parity check FAILED.");
        }

        scanner.close();
    }
}
```

```
Enter binary data (without parity bit): 1011
Enter parity type (even/odd): even
Data with even parity bit: 10111
Parity check PASSED.
```

```
Enter binary data (without parity bit): 1011
Enter parity type (even/odd): odd
Data with odd parity bit: 10110
Parity check PASSED.
```

```
Enter binary data (without parity bit): 1100
Enter parity type (even/odd): even
Data with even parity bit: 11000
Parity check PASSED.
```

```
Enter binary data (without parity bit): 1100
Enter parity type (even/odd): odd
Data with odd parity bit: 11001
Parity check PASSED.
```

Date: 16/09/2025

Lab Practical #15:

Implementation of parity bit check Using C/Java language with example.

Practical Assignment #15:

C/Java Program: Implementation of Bit stuffing Using C/Java language.

```
import java.util.Scanner;

public class BitStuffing {

   public static String stuffBits(String data) {
      StringBuilder stuffed = new StringBuilder();
      int count = 0;

      for (char bit : data.toCharArray()) {
         stuffed.append(bit);a
         if (bit == '1') {
            count++;
            if (count == 5) {
               stuffed.append('0');
               count = 0;
            }
         } else {
            count = 0;
         }
      }

      return stuffed.toString();
   }

   public static String destuffBits(String stuffedData) {
      StringBuilder destuffed = new StringBuilder();
      int count = 0;
```

Date: 16/09/2025

```
        for (int i = 0; i < stuffedData.length(); i++) {
            char bit = stuffedData.charAt(i);
            destuffed.append(bit);

            if (bit == '1') {
                count++;
                if (count == 5 && i + 1 < stuffedData.length() && stuffedData.charAt(i +
1) == '0') {
                    i++;
                    count = 0;
                }
            } else {
                count = 0;
            }
        }

        return destuffed.toString();
    }
```

Date: 16/09/2025

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter binary data: ");
        String data = scanner.nextLine();

        String stuffed = stuffBits(data);
        System.out.println("Stuffed Data:   " + stuffed);

        String destuffed = destuffBits(stuffed);
        System.out.println("Destuffed Data: " + destuffed);

        scanner.close();
    }
}
```

1. **Enter the binary data: 011111101111110**

   Bit-stuffed data: 01111101011111010

```
Enter binary data: 011111101111110
Stuffed Data:   01111101011111010
Destuffed Data: 011111101111110
```

2. **Enter the binary data: 111110111111**

   Bit-stuffed data: 1 1 1 1 1 0 0 1 1 1 1 1 0 1

```
Enter binary data: 011111101111110
Stuffed Data:   01111101011111010
Destuffed Data: 011111101111110
```