CRC-CCITT 16

```
import java.util.Scanner;
public class CRC {
   private static final int polynomial = 0x1021; // CRC-CCITT polynomial
   public static String calculateCRC(String data) {
      int crc = 0xFFFF; // Initial CRC value
      for (char c : data.toCharArray()) {
         int byteValue = (int) c;
         crc ^= (byteValue << 8) & 0xFFFF;
         for (int i = 0; i < 8; i++) {
            if ((crc & 0x8000) != 0) {
               crc = (crc << 1) ^ polynomial;
            } else {
               crc <<= 1;
            }
            crc &= 0xFFFF; // Ensure it's a 16-bit value
         }
      }
      return Integer.toString(crc);
   }
   public static boolean validateCRC(String receivedData, String receivedCRC) {
      int crc = Integer.parseInt(receivedCRC);
      for (char c : receivedData.toCharArray()) {
         int byteValue = (int) c;
         crc ^= (byteValue << 8) & 0xFFFF;
         for (int i = 0; i < 8; i++) {
            if ((crc & 0x8000) != 0) {
               crc = (crc << 1) ^ polynomial;
            } else {
               crc <<= 1;
            }
            crc &= 0xFFFF; // Ensure it's a 16-bit value
         }
      }
      return crc == 0;
   }
   public static void main(String[] args) {
      Scanner scanner = new java.util.Scanner(System.in);
      // Sender side
      System.out.print("Enter the data for CRC calculation: ");
      String inputData = scanner.nextLine().trim();
      String crcChecksum = calculateCRC(inputData);
      String dataWithCRC = inputData + crcChecksum;
      System.out.println("Transmitting data with CRC: " + dataWithCRC);
      // Receiver side
      System.out.print("Enter the received data (message + CRC): ");
      String receivedData = scanner.nextLine().trim();
      String receivedMessage = receivedData.substring(0, receivedData.length() - 5);
```

```
        String receivedCRCString = receivedData.substring(receivedData.length() - 5);
        boolean isValid = validateCRC(receivedMessage, receivedCRCString);
        if (isValid) {
            System.out.println("CRC Check: Data is intact. Received message: " +
receivedMessage);
        } else {
            System.out.println("CRC Check: Data is corrupted. Discarding the message.");
        }
        scanner.close();
    }
}
```

Bellman Ford Algorithm:

```
import java.util.Arrays;
import java.util.Scanner;
public class BellmanFord {
    private static int N;
    private static int[][] graph;
    public static void bellmanFord(int src) {
        int[] dist = new int[N];
        Arrays.fill(dist, Integer.MAX_VALUE);
        dist[src] = 0;
        for (int i = 1; i < N; i++) {
            for (int u = 0; u < N; u++) {
                for (int v = 0; v < N; v++) {
                    if (graph[u][v] != 0 && dist[u] != Integer.MAX_VALUE && dist[u] + graph[u][v]
< dist[v]) {
                        dist[v] = dist[u] + graph[u][v];
                    }
                }
            }
        }
        for (int u = 0; u < N; u++) {
            for (int v = 0; v < N; v++) {
                if (graph[u][v] != 0 && dist[u] != Integer.MAX_VALUE && dist[u] + graph[u][v] <
dist[v]) {
                    System.out.println("Negative weight cycle detected.");
                    return;
                }
            }
        }
        printSolution(dist);
    }
    public static void printSolution(int[] dist) {
        System.out.println("Vertex \t Distance from Source");
        for (int i = 0; i < N; i++) {
            System.out.println((i + 1) + "\t\t" + dist[i]);
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of Vertices : ");
        N = sc.nextInt();
        System.out.println("Enter the Weight Matrix of Graph");
        graph = new int[N][N];
        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++)
                graph[i][j] = sc.nextInt();
        System.out.print("Enter the Source Vertex : ");
        int source = sc.nextInt();
        bellmanFord(source - 1);
    }
}
```

Token Bucket Algorithm

```
import java.util.Random;
import java.util.Scanner;
public class TokenBucket {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
        System.out.print("Enter The Number of Packets : ");
        int n = scanner.nextInt();
        int tokens = 0;
        int bsize = 0;
        System.out.print("Enter The Bucket Size : ");
        bsize = scanner.nextInt();
        tokens = bsize;
        int outrate = random.nextInt(bsize - 1) + 1;
        int[] packets = new int[n];
        System.out.println("Enter The Packet Sizes in Order ");
        for (int i = 0; i < n; i++) {
            packets[i] = scanner.nextInt();
        }
        int i = 0, cycle = 0, remains = 0, sent = 0;
        boolean flag = false;
        System.out.println("Cycle\tPackets\tSent\tRemains");
        while (true) {
            cycle++;
            tokens = bsize - remains;
            if (packets[i] <= tokens) {
                if (remains + packets[i] <= outrate) {
                    sent = remains + packets[i];
                    remains = 0;
                } else {
                    remains += (packets[i] - outrate);
                    sent = outrate;
                }
                if (!flag) {
                    System.out.println(cycle + "\t" + packets[i] + "\t" + sent + "\t" + remains);
                    packets[i] = 0;
                } else
                    System.out.println(cycle + "\t---\t" + sent + "\t" + remains);
            } else {
                remains = bsize;
                if (remains <= outrate) {
                    sent = remains;
                    remains = 0;
                } else {
                    remains -= outrate;
                    sent = outrate;
                }
                if (!flag) {
                    System.out.println(cycle + "\t" + packets[i] + "\t" + sent + "\t" + remains);
                    packets[i] -= tokens;
                } else
                    System.out.println(cycle + "\t---\t" + sent + "\t" + remains);
            }
            if (packets[i] != 0)
                continue;
            else if (i == (packets.length - 1)) {
                flag = true;
                if (remains == 0) {
                    break;
                }
            } else
                i++;
        }
        scanner.close();
      }
}
```