Program 10: Write a program for congestion control using leaky bucket algorithm.

```
import java.util.*;  // Import Scanner class for user input

class LeakyBucket {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);  // Create Scanner object for input

        // Get the size of the bucket
        System.out.print("Enter bucket size: ");
        int bucketSize = sc.nextInt();

        // Get the constant output rate
        System.out.print("Enter output rate: ");
        int outRate = sc.nextInt();

        // Get the number of seconds for which packet data will be entered
        System.out.print("Enter number of seconds: ");
        int n = sc.nextInt();

        // Array to store incoming packets for each second
        int[] packets = new int[n];
        System.out.println("Enter incoming packets each second:");
        for (int i = 0; i < n; i++)
            packets[i] = sc.nextInt();  // Read incoming packets for second i

        int bucket = 0;  // Current number of packets in the bucket

        // Display table header
        System.out.println("\nTime\tIncoming\tSent\tDropped\tRemaining");

        // Process each second's incoming packets
        for (int i = 0; i < n; i++) {
            bucket += packets[i];  // Add incoming packets to bucket
            int dropped = 0;       // Initialize dropped packets count

            // If bucket overflows, drop extra packets
            if (bucket > bucketSize) {
```

```
            dropped = bucket - bucketSize;  // Calculate how many packets dropped
            bucket = bucketSize;          // Keep bucket at max capacity
        }

        // Send packets at constant output rate
        int sent = Math.min(bucket, outRate);
        bucket -= sent;  // Reduce sent packets from bucket

        // Display time, incoming, sent, dropped, and remaining packets
        System.out.println((i + 1) + "\t" + packets[i] + "\t\t" + sent + "\t" + dropped + "\t" +
bucket);
      }

      // Continue sending remaining packets after all inputs are processed
      while (bucket > 0) {
        int sent = Math.min(bucket, outRate);
        bucket -= sent;
        System.out.println("-\t0\t\t" + sent + "\t0\t" + bucket);
      }

      sc.close();  // Close the scanner
   }
}
```