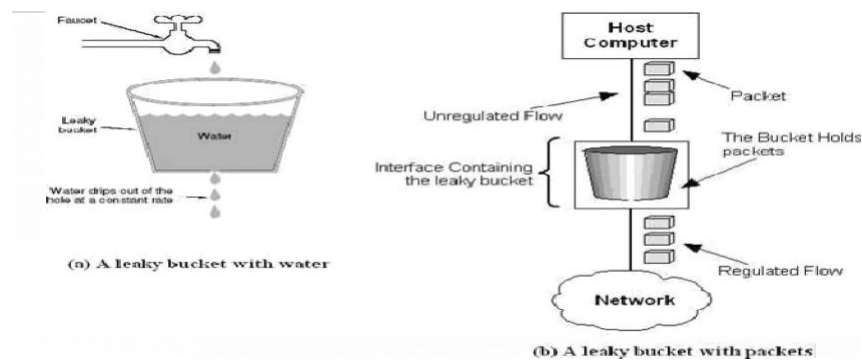

3. Write a program for congestion control using leaky bucket algorithm.

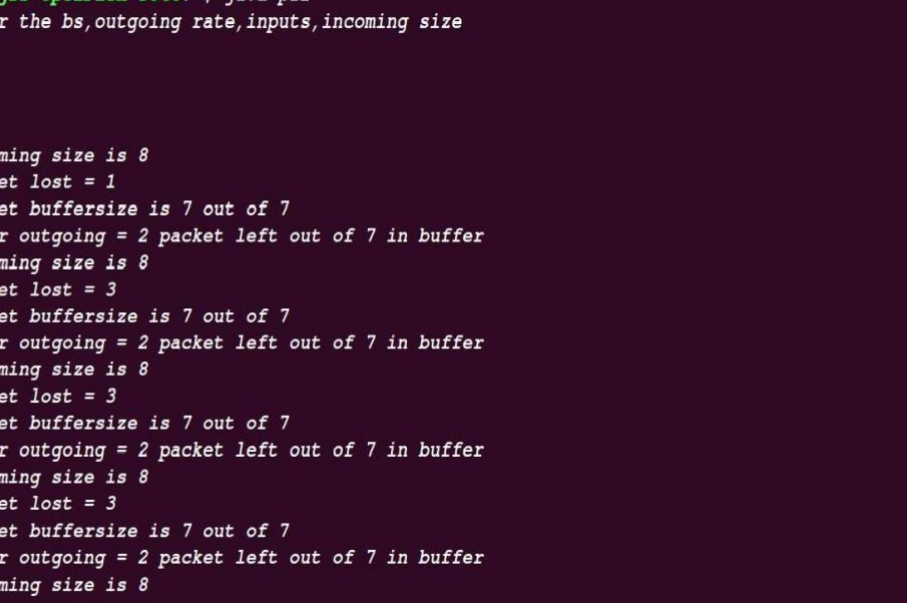
The main concept of the leaky bucket algorithm is that the output data flow remains constant despite the variant input traffic, such as the water flow in a bucket with a small hole at the bottom. In case the bucket contains water (or packets) then the output flow follows a constant rate, while if the bucket is full any additional load will be lost because of spillover. In a similar way if the bucket is empty the output will be zero. From network perspective, leaky bucket consists of a finite queue (bucket) where all the incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. In order to regulate the output flow, leaky bucket transmits one packet from the queue in a fixed time (e.g. at every clock tick). In the following figure we can notice the main rationale of leaky bucket algorithm, for both the two approaches (e.g. leaky bucket with water (a) and with packets (b)).



While leaky bucket eliminates completely bursty traffic by regulating the incoming data flow its main drawback is that it drops packets if the bucket is full. Also, it doesn't take into account the idle process of the sender which means that if the host doesn't transmit data for some time the bucket becomes empty without permitting the transmission of any packet.

Source Code:

```
import java.util.Scanner;
public class p12
{
    public static void main(String[] args) throws InterruptedException
    {
        Scanner in=new Scanner(System.in);
        int n,incoming,outgoing,bs,s=0;
        System.out.println("enter the bs,outgoing rate,inputs,incoming size");
        bs=in.nextInt();
        outgoing=in.nextInt();
        n=in.nextInt();
        incoming=in.nextInt();
        while(n!=0)
        {
            System.out.println("incoming size is"+incoming);
            if(incoming<=(bs-s))
            {
                s+=incoming;
                System.out.println("bucket buffer size is"+s+"out of"+bs);
            }
            else
            {
                System.out.println("packet lost="+incoming-(bs-s));
                s=bs;
                System.out.println("bucket buffersize is"+s+"out of"+bs);
            }
            s-=outgoing;
            System.out.println("after outgoing="+s+"packet left out of"+bs+"in buffer");
            n--;
            Thread.sleep(3000);
        }
        in.close();
    }
}
```



Terminal window showing the execution of a Java program. The user runs `javac p12.java` and `java p12`. The program outputs network statistics, including incoming size, packet loss, and bucket buffersize.

```
jss@jss-OptiPlex-3046: ~$ javac p12.java
jss@jss-OptiPlex-3046: ~$ java p12
enter the bs,outgoing rate,inputs,incoming size
7
5
10
8
incoming size is 8
packet lost = 1
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
incoming size is 8
packet lost = 3
bucket buffersize is 7 out of 7
after outgoing = 2 packet left out of 7 in buffer
```

[illegible]