**Problem Statement:**

Write a program for congestion control using Leaky bucket algorithm.

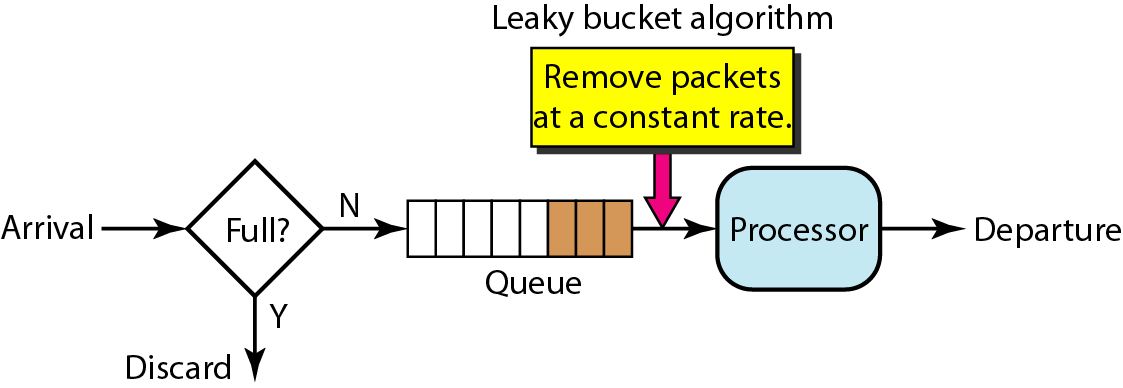
**Theory:**

Leaky bucket (proposed by Jonathan S. Turner, 1986) is a traffic shaping algorithm. Each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more process are already queued, the new packet is unceremoniously discarded.

The host is allowed to put one packet per clock tick onto the network. This mechanism turns an uneven flow of packet from the user process inside the host into an even flow of packet onto the network, smoothing out bursts and greatly reducing the chances of congestion.

**Algorithm**

1. The leaky bucket consists of a finite queue.
2. When a packet arrives, if there is room on the queue it is appended to the queue; otherwise, it is discarded.
3. At every clock tick, one packet is transmitted



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* File Name : LeakyBucket.c

\*

\* Description : A program to implement Leaky Bucket Algorithm

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "stdafx.h"

#include<stdio.h>

#include<stdlib.h>

#define BUCKETSIZE 250

#define OUTRATE 25

struct

{

int arrtime;

int weight;

}packet[15];

void read\_packets(int n)

{

int i;

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

{

printf("Enter arrival time:");

scanf("%d",&packet[i].arrtime);

printf("Enter the packet size:");

scanf("%d",&packet[i].weight);

}

}

// leaky bucket algorithm

void lky\_buckNextet()

{

int excess=BUCKETSIZE;

int i,j=0,rem=0;

// receive packets for 30 msec

for(i=0;i<=30;i++)

{

if(packet[j].arrtime==i)

{

if(packet[j].weight<=excess)

{

rem=packet[j].weight+rem;

excess=excess-packet[j].weight;

printf("At time=%d: packet %d inserted into bucket,",i,j+1);

printf(" remaining bucket size= %d\n",excess);

j=j+1;

}

else

{

printf("At time = %d: packet %d discarded,",i,j+1);

printf("Packet size is more than buffer size\n");

j=j+1;

}

}

// send packets every 5 msec

if((i%5)==0)

{

if(rem>=OUTRATE)

{

rem=rem-OUTRATE;

excess=excess+OUTRATE;

printf("At time = %d : 25 Kbytes are transfered ",i);

printf("Free available space in the bucket=%d\n",excess);

}

else if(rem>0)

{

excess=excess+rem;

printf("At time = %d : %d Kbytes are transferred ",i,rem);

printf("Free available space in the bucket=%d\n",excess);

rem=0;

}

}

}//for

// Empty the bucket

while(rem != 0)

{

if(rem < OUTRATE)

{

excess=excess+rem;

printf("At time = %d : %d Kbytes are transfered ",i++,rem);

printf("Free available space in the bucket=%d\n",excess);

break;

}

rem=rem-OUTRATE;

excess=excess + OUTRATE;

printf("At time = %d : %d Kbytes are transfered ",i++,OUTRATE);

printf("Free available space in the bucket=%d\n",excess);

}

printf("Bucket is empty");

}

int main()

{

int n;

printf("Enter the number of packets:");

scanf("%d",&n);

read\_packets(n);

lky\_bucket();

return 0;

}