Fwd:

1 message

Chethan P P <ppchethan1@gmail.com> Mon, 4 Mar, 2024 at 10:17
To: gagan.kulal619@gmail.com

---------- Forwarded message ---------
From: **Chethan P P** <ppchethan1@gmail.com>
Date: Mon, 4 Mar, 2024, 10:08 am
Subject:
To: <ashwinvsappu87@gmail.com>

[03/03, 6:50 pm] Chethan Podnolana: import java.io.*;

```
class Lab1 {
  public static void main(String args[]) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter Generator:");
    String gen = br.readLine();
    System.out.println("Enter Data:");
    String data = br.readLine();
    String code = data;
    while (code.length() < (data.length() + gen.length() - 1))
      code = code + "0";
    code = data + div(code, gen);
    System.out.println("The transmitted Code Word is:" + code);
    System.out.println("Please enter the received Code Word: ");
    String rec = br.readLine();
    if (Integer.parseInt(div(rec, gen)) == 0)
      System.out.println("The received code word contains no errors.");
    else
      System.out.println("The received code word contains errors.");
  }
  static String div(String num1, String num2) {
    int pointer = num2.length();
    String result = num1.substring(0, pointer);
    String remainder = "";
    for (int i = 0; i < num2.length(); i++) {
      if (result.charAt(i) == num2.charAt(i))
        remainder += "0";
      else
        remainder += "1";
    }
    while (pointer < num1.length()) {
      if (remainder.charAt(0) == '0')
        remainder = remainder.substring(1, remainder.length());
      remainder = remainder + String.valueOf(num1.charAt(pointer));
      pointer++;
    }
    result = remainder;
    remainder = "";
    for (int i = 0; i < num2.length(); i++)
      if (result.charAt(i) == num2.charAt(i))
        remainder += "0";
      else
        remainder += "1";
    return remainder.substring(1, remainder.length());
  }
}
```

[03/03, 6:50 pm] Chethan Podnolana: import java.util.Scanner;

```
public class Leaky_test{
public static void main(String[] args) {
    int n,rate,size;
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter the number of packets:");
    n=sc.nextInt();
    System.out.println("Enter the output rate of the bucket:");
    rate =sc.nextInt();
    System.out.println("Enter the bucket size:");
    size=sc.nextInt();
    System.out.println("Enter the packet size:");
    int[]packets=new int[n];
    for(int i=0;i<n;i++){
      packets[i]=sc.nextInt();
    }
    System.out.println("\nClock\tSize\tStatus\t\tSent\tRemaining");
    int rem=0,sent=0;
    for(int i=0;i<n;i++){
      try{
        Thread.sleep(1500);
      }
      catch(InterruptedException e){
        e.printStackTrace();
      }
      System.out.print((i+1)+"\t"+packets[i]+"\t");
      if ((rem+packets[i])>size) {
        if ((rem>=rate)) {
          rem-=rate;
          sent=rate;
        }
        else{
          sent=rem;
          rem=0;
        }
        System.out.println("Dropped\t\t"+sent+"\t"+rem);
      }
      else{
        rem+=packets[i];
        if (rem>=rate) {
          rem-=rate;
          sent=rate;
        }
        else{
          sent=rem;
          rem=0;
        }
        System.out.print("Accepted\t"+sent+"\t"+rem+"\n");
      }
    }
  }
}
```

[03/03, 6:50 pm] Chethan Podnolana: import java.util.*;

```
public class BellmanFord{
  private int D[];
  private int num_ver;
  public static final int MAX_VALUE = 999;

  public BellmanFord(int num_ver){
    this.num_ver = num_ver;
    D = new int[num_ver+1];
  }

  public static void main(String[] args){
    int num_ver=0;
    int source;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the number of Vertices: ");
    num_ver = sc.nextInt();
    int A[][] = new int[num_ver + 1][num_ver + 1];
    System.out.println("Enter the adjacency matrix: ");
    for(int sn = 1; sn <= num_ver; sn++){
      for(int dn = 1;dn<=num_ver;dn++){
```

```
                A[sn][dn]=sc.nextInt();
                if(sn==dn){
                    A[sn][dn]=0;
                    continue;
                }
                if(A[sn][dn]==0){
                    A[sn][dn]=MAX_VALUE;
                }
            }
        }
        System.out.println("Enter Source Vertex");
        source = sc.nextInt();
        BellmanFord b = new BellmanFord(num_ver);
        b.BellmanFordEvaluation(source,A);
        sc.close();
    }

    public void BellmanFordEvaluation(int source, int A[][]){
        for(int node = 1; node<=num_ver;node++){
            D[node] = MAX_VALUE;
        }
        D[source]=0;
        for(int node=1;node<=num_ver-1;node++){
            for(int sn = 1; sn<=num_ver;sn++){
                for(int dn=1;dn<=num_ver;dn++){
                    if(A[sn][dn]!=MAX_VALUE){
                        if(D[dn]>D[sn]+A[sn][dn])
                        D[dn]=D[sn]+A[sn][dn];
                    }
                }
            }
        }
        for(int sn = 1;sn<=num_ver;sn++){
            for(int dn = 1; dn<= num_ver;dn++){
                if(A[sn][dn]!=MAX_VALUE){
                    if(D[dn]>D[sn]+A[sn][dn]){
                        if(D[dn]>D[sn]+A[sn][dn])
                        System.out.println("The Graph contains negative edge cycle");
                    }
                }
            }
        }
        for(int vertex = 1; vertex <= num_ver;vertex++){
            System.out.println("Distance of Source"+ source +" to "+vertex+" is "+D[vertex]);
        }
    }
}
```