```
import java.net.*;
import java.util.*;

public class EXP9 {
   public static void main(String[] a) {
      Scanner s = new Scanner(System.in);
      int n;
      do {
         System.out.println("1. DNS\n2. Exit\nEnter your choice:");
         n = s.nextInt();
         if (n == 1) {
            System.out.println("Enter the hostname:");
            String h = s.next();
            try {
               InetAddress ad = InetAddress.getByName(h);
               System.out.println("Host Name: " + ad.getHostName() + "\nIP Address: " +
ad.getHostAddress());
            } catch (UnknownHostException e) {
               System.out.println("Unable to find IP address. Please enter a valid hostname.");
            }
         }
      } while (n == 1);
      s.close();
   }
}
```

```
import java.util.Scanner;

class EXP8 {
    public static void main(String[] a) {
        Scanner s = new Scanner(System.in);
        // User inputs
        System.out.print("Enter bucket size: ");
        int B = s.nextInt(); // Bucket size
        System.out.print("Enter number of queries: ");
        int Q = s.nextInt(); // Number of queries
        System.out.print("Enter input packet size: ");
        int I = s.nextInt(); // Input packet size
        System.out.print("Enter output packet size: ");
        int O = s.nextInt(); // Output packet size
        int S = 0; // Initial storage in the bucket
        for (int i = 0; i < Q; i++) {
            int L = B - S; // Space left
            if (I <= L) {
                S += I; // Store packets if space is available
            } else {
                System.out.println("Packet loss = " + (I - L)); // Calculate packet loss
            }
            // Show current buffer status
            System.out.println("Buffer size = " + S + " out of bucket size = " + B);
            S -= O; // Remove packets from the bucket
            if (S < 0) S = 0; // Avoid negative storage
        }
       // Simple output explanation
        System.out.println("Final buffer size = " + S + " out of bucket size = " + B);
        s.close();
    }
}
```

```
import java.util.Scanner;
public class EXP7 {
    static final int I = 9999;
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter number of nodes: ");
        int n = s.nextInt();
        s.nextLine();
        char[] a = new char[n];
        for (int i = 0; i < n; i++) {
            a[i] = (char) ('A' + i);
        }
        int[][] d = new int[n][n];
        int[][] h = new int[n][n];
        System.out.println("Enter the distances (9999 if no connection): ");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print("Distance from " + a[i] + " to " + a[j] + ": ");
                d[i][j] = s.nextInt();
                if (i == j) {
                    d[i][j] = 0;
                }
                h[i][j] = (d[i][j] == I) ? -1 : j;
            }
        }
        System.out.println("\nInitial Tables:");
        p(a, d, h);
        // Perform updates n-2 times
        for (int x = 0; x < n - 2; x++) {
            for (int i = 0; i < n; i++) { // for each node
                for (int j = 0; j < n; j++) { // check distance to every other node
                    for (int k = 0; k < n; k++) { // compare via every other node
```

```
                    if (d[i][k] + d[k][j] < d[i][j]) {
                        d[i][j] = d[i][k] + d[k][j];
                        h[i][j] = h[i][k];
                    }
                }
            }
        }
        System.out.println("\nAfter iteration " + (x + 1) + ":");
        p(a, d, h);
    }
    System.out.println("\nFinal Tables:");
    p(a, d, h);
  }
  private static void p(char[] a, int[][] d, int[][] h) {
    int n = a.length;
    for (int i = 0; i < n; i++) {
      System.out.println("Table for node " + a[i] + ":");
      System.out.println("Dst\tDis\tHop");
        for (int j = 0; j < n; j++) {
         if (d[i][j] == I) {
            System.out.println(a[j] + "\tINF\t-");
         } else {
            System.out.println(a[j] + "\t" + d[i][j] + "\t" + a[h[i][j]]);
         }
      }
      System.out.println();
    }
  }
}
```

```
import java.util.*;
public class Exp6 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        // Get the number of nodes
        System.out.print("Enter number of nodes: ");
        int n = s.nextInt();
        // Initialize distance matrix
        int[][] g = new int[n][n];
        System.out.println("Enter the distance matrix (use 9999 for infinity):");
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                g[i][j] = s.nextInt();
        // Get the source node
        System.out.print("Enter source node: ");
        int src = s.nextInt();
        // Dijkstra's algorithm
        int[] d = new int[n];
        boolean[] v = new boolean[n];
        Arrays.fill(d, 9999); // Initialize distances as infinity
        d[src] = 0;
        for (int c = 0; c < n - 1; c++) {
            int u = -1;
            for (int i = 0; i < n; i++)
                if (!v[i] && (u == -1 || d[i] < d[u]))
                    u = i;
            v[u] = true;
            for (int j = 0; j < n; j++)
                if (!v[j] && g[u][j] != 9999 && d[u] + g[u][j] < d[j])
                    d[j] = d[u] + g[u][j];
        }
        // Print shortest distances
```

```
        System.out.println("Shortest distances from node " + src + ":");
        for (int i = 0; i < n; i++)
            System.out.println("To " + i + " - " + (d[i] == 9999 ? "Infinity" : d[i]));
        s.close();
    }
}

import java.io.*;

class EXP05 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter Generator: ");
        String g = br.readLine();
        System.out.print("Enter Data: ");
        String d = br.readLine();
        String c = d + div(d + "0".repeat(g.length() - 1), g);
        System.out.println("Transmitted Code Word: " + c);

        System.out.print("Enter Received Code Word: ");
        String r = br.readLine();
        System.out.println(div(r, g).contains("1") ? "Errors in received code." : "No errors in received
code.");
    }

    static String div(String n, String g) {
        String rem = n.substring(0, g.length());
        for (int i = g.length(); i <= n.length(); i++) {
            rem = rem.charAt(0) == '0' ? rem.substring(1) : xor(rem, g).substring(1);
            if (i < n.length()) rem += n.charAt(i);
        }
        return rem;
    }
```

```
    static String xor(String a, String b) {
        StringBuilder res = new StringBuilder();
        for (int i = 0; i < b.length(); i++)
            res.append(a.charAt(i) == b.charAt(i) ? '0' : '1');
        return res.toString();
    }
}
```