

```

import java.util.Scanner;

public class TracciaB {

    /**
     * Si scriva un metodo verificaVettore che riceve in ingresso due
     * vettori di interi v e w ed un intero k, e restituisce un valore booleano. In
     * particolare restituisce true se per ogni elemento v[i] appartenente a
     * v
     * esiste in w un elemento w[j] tale che v[i]+w[j] = k, false
     * altrimenti. Ad
     * esempio, nel caso in cui v= [ 1, 5, 3, 2], w = [ 5, 12, 7, 9, 8, 2] e
     * k =
     * 10 , il metodo restituisce true.
     */
    public static boolean verificaVettore(int[] v, int w[], int k) {
        for (int i = 0; i <= v.length; i++) {
            boolean trovato = false;
            for (int j = 0; j < w.length && !trovato; j++)
                if (v[i] + w[j] == k)
                    trovato = true;
            if (!trovato)
                return false;
        }
        return true;
    }

    /**
     * 1. un metodo verificaMatrice che riceve in ingresso una matrice M e
     * restituisce in output un booleano. In particolare il metodo
     * restituisce
     * true se la somma degli elementi di ciascuna riga di M è identica,
     * false
     * altrimenti
     */
    public static boolean verificaMatrice(int[][] m) {
        int sommaPrima = 0;
        for (int j = 0; j < m[0].length; j++) {
            sommaPrima += m[0][j];
        }
        for (int i = 1; i < m.length; i++) {
            int somma = 0;
            for (int j = 0; j < m[0].length; j++)
                somma += m[i][j];

            if (somma != sommaPrima)
                return false;
        }
        return true;
    }

    /**
     * 2. un metodo estraiArray che riceve in ingresso una matrice quadrata
     * M e
     * restituisce in output un array V contenente gli elementi di M al di
     * sotto
     * della diagonale principale senza duplicati
     */
    public static int[] estraiArray(int[][] m) {
        int[] temp = new int[(m.length * m.length - m.length) / 2];
    }
}

```

```

                                TracciaB
int k = 0;
for (int i = 0; i < m.length; i++)
    for (int j = 0; j < i; j++) {
        boolean presente = false;
        for (int l = 0; l < k && !presente; l++)
            if (temp[l] == m[i][j])
                presente = true;
        if (!presente) {
            temp[k] = m[i][j];
            k++;
        }
    }

int[] ris = new int[k];
System.arraycopy(temp, 0, ris, 0, k);
return ris;
}

/*
 * 3. un metodo azzerMatrice che riceve in ingresso una matrice
quadrata M
 * di dimensione n e un array V e restituisce in output la matrice H
 * ottenuta dalle ultime n/2 colonne di M e contenente i corrispondenti
 * elementi di M se non presenti nell'array V, 0 altrimenti
 */
public static int[][] azzerMatrice(int[][] m, int[] v) {
    int[][] ris = new int[m.length][m.length / 2];
    int c = m.length - ris[0].length;
    for (int i = 0; i < ris.length; i++) {
        for (int j = 0; j < ris[0].length; j++) {
            if (presente(v, m[i][c + j]))
                ris[i][j] = 0;
            else
                ris[i][j] = m[i][c + j];
        }
    }
    return ris;
}

static boolean presente(int[] a, int k) {
    for (int i = 0; i < a.length; i++)
        if (a[i] == k)
            return true;
    return false;
}

/*
 * 4. un metodo main che legge una matrice quadrata M di numeri interi e
 * dimensione dispari e provvede, invocando opportunamente i metodi
 * proposti, a realizzare i compiti di cui ai punti (1), (2) e (3)
 */
public static void main(String[] args) {
    Scanner console = new Scanner(System.in);
    int dim = 0;
    do {
        System.out.println("inserire dim matrice");
        dim = console.nextInt();
        if (dim <= 0) {
            System.out.println("la dim deve essere
positiva!!");
        }
    } while (dim <= 0);

    int[][] M = new int[dim][dim];

```

```

                                TracciaB
leggiMatrice(M);
boolean esito = verificaMatrice(M);
System.out.println("risultato invocazione verificaMatrice: " +
esito);

int[] v1 = estraiArray(M);
System.out.println("risultato invocazione estraiArray");
stampaVettore(v1);
int[] v= new int[4];
leggiVettore(v);
int[][] d = azzeraMatrice(M,v);
System.out.println("risultato invocazione azzera matrice");
stampaMatrice(d);

}

public static void stampaVettore(int[] v) {
    for (int i = 0; i < v.length; i++)

        System.out.print(v[i] + " ");

    System.out.println();

}

public static void leggiMatrice(int[][] m) {
    Scanner console = new Scanner(System.in);
    for (int i = 0; i < m.length; i++) {
        for (int j = 0; j < m[0].length; j++) {
            System.out.print("m[" + i + ", " + j + "]=");
            m[i][j] = console.nextInt();
        }
    }
}

public static void leggiVettore(int[] v) {
    Scanner console = new Scanner(System.in);
    for (int i = 0; i < v.length; i++) {
        System.out.print("v[" + i + "]=");
        v[i] = console.nextInt();
    }
}

public static void stampaMatrice(int[][] m) {
    for (int i = 0; i < m.length; i++) {
        for (int j = 0; j < m[0].length; j++) {
            System.out.print(m[i][j] + " ");
        }
        System.out.println();
    }
}
}

```