```java
public class SimulazioneConSoluzioni {
    /** ESERCIZIO 1.
     * Scrivere un metodo iterativo e1 con le seguenti
     * caratteristiche:
     * -) e1 ha un parametro formale di tipo matrice
     * bidimensionale di interi che puo' essere solo
     * quadrata, o nulla.
     * -) e1 restituisce true quando:
     *    a) la matrice non e' nulla e
     *    b) la somma degli elementi di ciascuna riga
     *    concide con la somma degli elementi della
     *    colonna corrispondente. */

    public static boolean e1(int[][] x) {
        boolean risultato = (x != null);
        if (risultato) {
            for (int i = 0; risultato && i < x.length; i++) {
                int r = 0, c = 0;
                for (int j = 0; j < x[i].length; j++)
                    r = r + x[i][j];
                for (int j = 0; j < x.length; j++)
                    c = c + x[j][i];
                risultato = r == c;
            }
        }
        return risultato;
    }
```

```java
    /** ESERCIZIO 2.
     * Scrivere un metodo ricorsivo dicotomico e2 con
     * le seguenti caratteristiche:
     * -) e2 ha un parametro formale di tipo matrice
     * bidimensionale di interi che puo' essere solo
     * quadrata, o nulla.
     * -) e2 restituisce true quando:
     *    a) la matrice non e' nulla e
     *    b) la somma degli elementi di ciascuna riga
     *    concide con la somma degli elementi della
     *    colonna corrispondente.
     * Per il calcolo della somma degli elementi in una
     * riga, definire un metodo ricorsivo sommaR
     * co-variante.
     * Per il calcolo della somma degli elementi in una
     * colonna, definire un metodo ricorsivo sommaC
     * contro-variante.                                 */

    public static boolean e2(int[][] x) {
        boolean risultato = (x != null);
        if (risultato) {
            risultato = x.length==0;
            if(!risultato)
                risultato = e2(x,0,x.length);
        }
        return risultato;
    }

    public static boolean e2(int[][] x, int l, int r) {
        if(l+1==r) {
            return sommaR(x[l],x[l].length) == sommaC(x,l,0);
        } else {
            int m = (l+r)/2;
            return e2(x,l,m) && e2(x,m,r);
        }
    }
```

```java


    public static int sommaR(int[] r, int i) {
        if(i==0) {
            return 0;
        } else {
            return r[i-1] + sommaR(r,i-1);
        }
    }

    public static int sommaC(int[][] c, int j, int i) {
        if(i==c.length) {
            return 0;
        } else {
            return c[i][j] + sommaC(c,j,i+1);
        }
    }
```

```java


/** ESERCIZIO 3.
 * Siano dati:
 * -) il metodo parity, qui sotto definito, da
 * applicare esclusivamente ad un parametro
 * attuale con almeno un elemento (a.length>=1)
 * -) il predicato P(i) seguente:
 *
 *  "Alla sua uscita, parity(a,i) rende vero
 *       'per ogni k.se 0<= k <= i,
 *        allora a[k]==(k%2==0)' ".
 *
 * 1) Scrivere il predicato P(0).
 * 2) Scrivere il predicato P(i-1) ==> P(i).
 * 3) Dimostrare che P(0) e' vero.
 * 4) Dimostrare che P(i-1) ==> P(i) e' vero,
 * ragionando induttivamente.                */
static void parity(boolean[] a, int i) {
    if (i < a.length) {
        if (i == 0)
            a[i] = true;
        else {
            parity(a, i - 1); //(A)
            a[i] = !a[i - 1]; //(B)
        }
    }
}
```

```


    /* Soluzione possibile.
     * 1) P(0) e'
     *     "Alla sua uscita, parity(a,0) rende vero
     *            'per ogni k.se 0<=k<=0,
     *            allora a[k]==(k%2==0)'"
     *
     * 2) P(i-1) ==> P(i) e'
     *    "Alla sua uscita, parity(a,i-1) rende vero
     *      'per ogni k.se 0<=k<=i-1,
     *                  allora a[k]==(k%2==0)'"
     *    ==> "Alla sua uscita, parity(a,i) rende vero
     *          'per ogni k.se 0<=k<=i,
     *                      allora a[k]==(k%2==0)'"
     *
     * 3) Osserviamo che:
     * --) per definizione, parity(a,0) esegue a[0]=true
     * e poi termina.
     * --) 0%2==0 e' vero.
     * Per l'unico valore di k nell'intervallo [0,0],
     * possiamo scrivere:
     *                   a[0] == true == (0%2==0)
     * da cui segue che P(0) e' vero.
     *
     * 4) Per ipotesi induttiva assumiamo P(i-1) vero:
     *
     *    "Alla sua uscita, parity(a,i-1) rende vero
     *      'per ogni k.se 0<= k<=i-1,
     *                  allora a[k]==(k%2==0)' "
     *
     * e' vero.
     *
     * --- Caso 1) Assumiamo che i-1 sia pari.
     * Quindi, per ipotesi, e' vera la congiunzione:
     *   a[0]==(0%2==0) && a[1]==(1%2==0)
     *    && a[2]==(2%2==0) &&...&& a[i-1]==((i-1)%2==0).
     * nel punto (A) che e' seguito dalla assegnazione
     * 'a[i] = !a[i-1]'. Siccome i-1 e' pari, allora i
```

```
     * e' dispari e possiamo scrivere:
     *
     *      a[i] == !a[i-1]
     *           == !((i-1)%2==0)
     *           == !true
     *           == false
     *           == (i%2==0).
     *
     * Nel punto (B) diventa vera la congiunzione:
     *
     * a[0]==(0%2==0) && a[1]==(1%2==0)
     *   && a[2]==(2%2==0) &&...&& a[i-1]==((i-1)%2==0)
     *   && a[i]==(i%2==0)
     *
     * che possiamo riassumere come:
     *
     *  "Alla sua uscita, parity(a,i) rende vero
     *    'per ogni k.se 0<= k<=i,
     *                allora a[k]==(k%2==0)' "
     *
     * che e' esattamente P(i).
     *
     * --- Caso 2) Assumiamo che i-1 sia dispari.
     * Quindi, per ipotesi, e' vera la congiunzione:
     *
     *  a[0]==(0%2==0) && a[1]==(1%2==0)
     *   && a[2]==(2%2==0) &&...&& a[i-1]==((i-1)%2==0)
     *
     * nel punto (A) che e' seguito dalla assegnazione
     * 'a[i] = !a[i-1]'. Siccome i-1 e' dispari, allora
     * i e' pari e possiamo scrivere:
     *
     *      a[i] == !a[i-1]
     *           == !((i-1)%2==0)
     *           == !false
     *           == true
     *           == (i%2==0).
     *
     * Nel punto (B) diventa vera la congiunzione:
```

```
    *
    *  a[0]==(0%2==0) && a[1]==(1%2==0)
    *    && a[2]==(2%2==0) &&...&& a[i-1]==((i-1)%2==0)
    *    && a[i]==(i%2==0)
    *
    * che possiamo riassumere come:
    *
    * "Alla sua uscita, parity(a,i) rende vero
    *   'per ogni k.se 0<= k<=i,
    *               allora a[k]==(k%2==0)' "
    *
    * che e' esattamente P(i).                        */
```

```java


    /** ESERCIZIO 4. Disegnare lo stato della memoria
     * immediatamente prima della disallocazione del
     * record di attivazione del metodo stack, quando i
     * ha valore 2. */
    static void stack(int[][] x, int i) {
        if (i < x.length) {
            int[] l = new int[x[i].length];
            l[i] = x[i][i] + 1;
            x[i] = l;
            stack(x, i + 1); // (B)
        }
    }

    public static void main(String[] args) {
        int[][] y = {{0,0}, {0,0}};
        stack(y, 0); // (A)
    }
}
```