

SimulazioneConSoluzioni.java

```
1 public class SimulazioneConSoluzioni {
2     /** ESERCIZIO 1.
3      * Scrivere un metodo iterativo el con le seguenti
4      * caratteristiche:
5      * -) el ha un parametro formale di tipo matrice
6      * bidimensionale di interi che puo' essere solo
7      * quadrata, o nulla.
8      * -) el restituisce true quando:
9      * a) la matrice non e' nulla e
10     * b) la somma degli elementi di ciascuna riga
11     * concide con la somma degli elementi della
12     * colonna corrispondente. */
13
14     public static boolean el(int[][] x) {
15         boolean risultato = (x != null);
16         if (risultato) {
17             for (int i = 0; risultato && i < x.length; i++) {
18                 int r = 0, c = 0;
19                 for (int j = 0; j < x[i].length; j++)
20                     r = r + x[i][j];
21                 for (int j = 0; j < x.length; j++)
22                     c = c + x[j][i];
23                 risultato = r == c;
24             }
25         }
26         return risultato;
27     }
28
29
30
31
32
33
34
35
36
37
38
39
```

SimulazioneConSoluzioni.java

```
40  /** ESERCIZIO 2.
41  * Scrivere un metodo ricorsivo dicotomico e2 con
42  * le seguenti caratteristiche:
43  * -) e2 ha un parametro formale di tipo matrice
44  * bidimensionale di interi che puo' essere solo
45  * quadrata, o nulla.
46  * -) e2 restituisce true quando:
47  * a) la matrice non e' nulla e
48  * b) la somma degli elementi di ciascuna riga
49  * concide con la somma degli elementi della
50  * colonna corrispondente.
51  * Per il calcolo della somma degli elementi in una
52  * riga, definire un metodo ricorsivo sommaR
53  * co-variante.
54  * Per il calcolo della somma degli elementi in una
55  * colonna, definire un metodo ricorsivo sommaC
56  * contro-variante. */
57
58  public static boolean e2(int[][] x) {
59      boolean risultato = (x != null);
60      if (risultato) {
61          risultato = x.length==0;
62          if(!risultato)
63              risultato = e2(x,0,x.length);
64      }
65      return risultato;
66  }
67
68  public static boolean e2(int[][] x, int l, int r) {
69      if(l+1==r) {
70          return sommaR(x[l],x[l].length) == sommaC(x,l,0);
71      } else {
72          int m = (l+r)/2;
73          return e2(x,l,m) && e2(x,m,r);
74      }
75  }
76
77
78
```

SimulazioneConSoluzioni.java

```
79
80
81 public static int sommaR(int[] r, int i) {
82     if(i==0) {
83         return 0;
84     } else {
85         return r[i-1] + sommaR(r,i-1);
86     }
87 }
88
89 public static int sommaC(int[][] c, int j, int i) {
90     if(i==c.length) {
91         return 0;
92     } else {
93         return c[i][j] + sommaC(c,j,i+1);
94     }
95 }
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
```

SimulazioneConSoluzioni.java

```
118
119
120  /** ESERCIZIO 3.
121   * Siano dati:
122   * -) il metodo parity, qui sotto definito, da
123   * applicare esclusivamente ad un parametro
124   * attuale con almeno un elemento (a.length>=1)
125   * -) il predicato P(i) seguente:
126   *
127   * "Alla sua uscita, parity(a,i) rende vero
128   * 'per ogni k.se 0<= k <= i,
129   * allora a[k]==(k%2==0)' ".
130   *
131   * 1) Scrivere il predicato P(0).
132   * 2) Scrivere il predicato P(i-1) ==> P(i).
133   * 3) Dimostrare che P(0) e' vero.
134   * 4) Dimostrare che P(i-1) ==> P(i) e' vero,
135   * ragionando induttivamente. */
136  static void parity(boolean[] a, int i) {
137      if (i < a.length) {
138          if (i == 0)
139              a[i] = true;
140          else {
141              parity(a, i - 1); //(A)
142              a[i] = !a[i - 1]; //(B)
143          }
144      }
145  }
146
147
148
149
150
151
152
153
154
155
156
```

SimulazioneConSoluzioni.java

```
157
158
159  /* Soluzione possibile.
160  * 1) P(0) e'
161  *    "Alla sua uscita, parity(a,0) rende vero
162  *      'per ogni k.se 0<=k<=0,
163  *        allora a[k]==(k%2==0)'"
164  *
165  * 2) P(i-1) ==> P(i) e'
166  *    "Alla sua uscita, parity(a,i-1) rende vero
167  *      'per ogni k.se 0<=k<=i-1,
168  *        allora a[k]==(k%2==0)'"
169  *    ==> "Alla sua uscita, parity(a,i) rende vero
170  *      'per ogni k.se 0<=k<=i,
171  *        allora a[k]==(k%2==0)'"
172  *
173  * 3) Osserviamo che:
174  *    --) per definizione, parity(a,0) esegue a[0]=true
175  *    e poi termina.
176  *    --) 0%2==0 e' vero.
177  *    Per l'unico valore di k nell'intervallo [0,0],
178  *    possiamo scrivere:
179  *      a[0] == true == (0%2==0)
180  *    da cui segue che P(0) e' vero.
181  *
182  * 4) Per ipotesi induttiva assumiamo P(i-1) vero:
183  *
184  *    "Alla sua uscita, parity(a,i-1) rende vero
185  *      'per ogni k.se 0<= k<=i-1,
186  *        allora a[k]==(k%2==0)' "
187  *
188  * e' vero.
189  *
190  * --- Caso 1) Assumiamo che i-1 sia pari.
191  * Quindi, per ipotesi, e' vera la congiunzione:
192  *   a[0]==(0%2==0) && a[1]==(1%2==0)
193  *   && a[2]==(2%2==0) &&...&& a[i-1]==((i-1)%2==0).
194  * nel punto (A) che e' seguito dalla assegnazione
195  * 'a[i] = !a[i-1]'. Siccome i-1 e' pari, allora i
```

SimulazioneConSoluzioni.java

```
196 * e' dispari e possiamo scrivere:
197 *
198 *     a[i] == !a[i-1]
199 *         == !((i-1)%2==0)
200 *         == !true
201 *         == false
202 *         == (i%2==0).
203 *
204 * Nel punto (B) diventa vera la congiunzione:
205 *
206 * a[0]==(0%2==0) && a[1]==(1%2==0)
207 * && a[2]==(2%2==0) &&...&& a[i-1]==((i-1)%2==0)
208 * && a[i]==(i%2==0)
209 *
210 * che possiamo riassumere come:
211 *
212 * "Alla sua uscita, parity(a,i) rende vero
213 * 'per ogni k.se 0<= k<=i,
214 *     allora a[k]==(k%2==0)' "
215 *
216 * che e' esattamente P(i).
217 *
218 * --- Caso 2) Assumiamo che i-1 sia dispari.
219 * Quindi, per ipotesi, e' vera la congiunzione:
220 *
221 * a[0]==(0%2==0) && a[1]==(1%2==0)
222 * && a[2]==(2%2==0) &&...&& a[i-1]==((i-1)%2==0)
223 *
224 * nel punto (A) che e' seguito dalla assegnazione
225 * 'a[i] = !a[i-1]'. Siccome i-1 e' dispari, allora
226 * i e' pari e possiamo scrivere:
227 *
228 *     a[i] == !a[i-1]
229 *         == !((i-1)%2==0)
230 *         == !false
231 *         == true
232 *         == (i%2==0).
233 *
234 * Nel punto (B) diventa vera la congiunzione:
```

SimulazioneConSoluzioni.java

```
235 *
236 *   a[0]==(0%2==0) && a[1]==(1%2==0)
237 *       && a[2]==(2%2==0) &&...&& a[i-1]==((i-1)%2==0)
238 *       && a[i]==(i%2==0)
239 *
240 * che possiamo riassumere come:
241 *
242 * "Alla sua uscita, parity(a,i) rende vero
243 *   'per ogni k.se 0<= k<=i,
244 *       allora a[k]==(k%2==0)' "
245 *
246 * che e' esattamente P(i).                                */
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
```

SimulazioneConSoluzioni.java

```
274
275
276  /** ESERCIZIO 4. Disegnare lo stato della memoria
277   * immediatamente prima della disallocazione del
278   * record di attivazione del metodo stack, quando i
279   * ha valore 2. */
280  static void stack(int[][] x, int i) {
281      if (i < x.length) {
282          int[] l = new int[x[i].length];
283          l[i] = x[i][i] + 1;
284          x[i] = l;
285          stack(x, i + 1); // (B)
286      }
287  }
288
289  public static void main(String[] args) {
290      int[][] y = {{0,0}, {0,0}};
291      stack(y, 0); // (A)
292  }
293 }
294
295
```