1. Scrivere un metodo ricorsivo che, dati un carattere `c` ed una stringa `s`, restituisce `true` se `c` occorre in `s`, `false` altrimenti.

```
public static boolean occorreCarRic (char c, String s) {
  return occorreCarRic(c,s,0);
}

public static boolean occorreCarRic (char c, String s, int i) {
  if (i == s.length())
    return false;
  if (s.charAt(i) == c)
    return true;
  return occorreCarRic(c,s,i+1);
}
```

2. Scrivere un metodo ricorsivo che, dati un carattere `c` ed una stringa `s`, restituisce il numero delle occorrenze di `c` in `s`.

   Soluzione con ricorsione annidata

```
public static int occorrenzeCarRic (char c, String s) {
  return occorrenzeCarRic(c,s,0);
}

public static int occorrenzeCarRic (char c, String s, int i) {
  if (i == s.length())
    return 0;
  if (s.charAt(i) == c)
    return 1 + occorrenzeCarRic(c,s,i+1);
  return occorrenzeCarRic(c,s,i+1);
}
```

   Soluzione con ricorsione in testa

```
public static int occorrenzeCarRic (char c, String s) {
  return occorrenzeCarRic(c,s,0,0);
}

public static int occorrenzeCarRic (char c, String s, int i, int k) {
  if (i == s.length())
    return k;
  if (s.charAt(i) == c)
    return occorrenzeCarRic(c,s,i+1,k+1);
  return occorrenzeCarRic(c,s,i+1,k);
}
```

3. Scrivere un metodo ricorsivo che, dati un carattere `c` ed una stringa `s`, restituisce la posizione della prima occorrenza di `c` in `s`, e `-1` se `c` non occorre in `s`.

```
public static int posizioneCarRic (char c, String s) {
  return posizioneCarRic(c,s,0);
}

public static int posizioneCarRic (char c, String s, int i) {
  if (i == s.length())
    return -1;
  if (s.charAt(i) == c)
    return i;
  return posizioneCarRic(c,s,i+1);
}
```

4. (*Esame scritto del 10/12/2001, una versione dell'Esercizio 5*.)
Scrivere un metodo ricorsivo in linguaggio Java che, dato un array di interi, ne restituisca il valore minimo.

Presentiamo due soluzioni. Nella prima si utilizza il metodo `min` della classe `Math` (che restituisce il minimo di due numeri dati in ingresso) e la chiamata ricorsiva del metodo risulta *annidata*. Nella seconda soluzione il metodo ricorsivo necessita un parametro in piú per tenere il valore del minimo corrente, mentre la chiamata ricorsiva del metodo è *in testa*.

Soluzione 1:

```
public static int minimo(int[] a) {
  return minimo(a,0);
}

public static int minimo(int[] a, int i) {
  if (i == a.length-1)
     return a[i];
  return Math.min(a[i],minimo(a,i+1));
}
```

Soluzione 2:

```
public static int minimo (int[] a) {
  return minimo(a,1,a[0]);
}

public static int minimo (int[] a, int i, int min) {
  if (i == a.length)
    return min;
  if (a[i] < min)
    return minimo(a,i+1,a[i]);
  return minimo(a,i+1,min);
}
```

Esercizio
Scrivere un metodo ricorsivo che, dato un array di interi `a`, restituisce il valore massimo in `a`.

5. Scrivere un metodo ricorsivo che, dato un array `a` di interi, restituisce la somma degli elementi di `a`.

```
public static int sommatoria (int[] a) {
  return sommatoria(a,0,0);
}

public static int sommatoria (int[] a, int i, int sum) {
  if (i == a.length)
    return sum;
  return sommatoria(a,i+1,sum+a[i]);
}
```

Esercizio
Scrivere un metodo ricorsivo che, dato un array di interi `a`, restituisce il prodotto degli elementi di `a`.

6. Scrivere un metodo ricorsivo che, dato un array bidimensionale `a` di interi, restituisce la somma degli elementi di `a`.

```
public static int sommatoria (int[][] a) {
  return sommatoria(a,0,0,0);
}

public static int sommatoria (int[][] a, int i, int j, int sum) {
  if (i == a.length)
    return sum;
  if (j == a[i].length)
    return sommatoria(a,i+1,0,sum);
  return sommatoria(a,i,j+1,sum+a[i][j]);
}
```