1. (*Esame scritto del 14/7/2004, Esercizio 1.*)
Si consideri il seguente frammento di codice in Java:

```
String[][] a = b;
int i = 0, j = 0, n = 0;
boolean p = true;
for (i=0; i<a.length; i++) {
  j = 0;
  p = true;
  while (j<a[i].length-1 && p) {
    if (a[i][j].charAt(0) == a[i][j+1].charAt(0)) {
      n++;
      j++;
    }
    else p = false;
  }
}
```

Determinare il valore finale delle variabili `i`,`j`, `n` e `p` per i seguenti valori dell'array `b`:
1) {{"aba","ac","pqr","afga","akk"},{"oops","opla","oh"}};
2) {{"bye","hi","hello"},{"help","aid","aiuto"},{"spqr","sport","copy"}}.

Soluzione
1) `i=2`, `j=2`, `n=3`, `p=true`;
2) `i=3`, `j=1`, `n=1`, `p=false`.

Alcuni esercizi sulla definizione di metodi che restituiscono array

2. (*Esame scritto del 4/11/2002, una versione dell'Esercizio 3.*)
Scrivere un metodo che, dato un array `a` di interi, restituisce l'array con gli elementi di `a` rovesciati. Ad esempio, dato `a = {5,-7,3,12,1}`, il metodo restituisce l'array `{1,12,3,-7,5}`.

```
public static int[] reverse (int[] a) {
  int[] b = new int[a.length];
  for (int i=0; i<a.length; i++)
   b[i] = a[a.length-1-i];
  return b;
}
```

3. (*Esame scritto del 4/11/2002, un'altra versione dell'Esercizio 3.*)
Scrivere un metodo che, dati due array `a` e `b` di interi, restituisce l'array in cui `b` è stato attaccato alla fine di `a`. Ad esempio, dati `a = {7,-3,2}` e `b = {-3,4,11,-21}`, il metodo restituisce l'array `{7,-3,2,-3,4,11,-21}`.

```
public static int[] append (int[] a, int[] b) {
  int[] c = new int[a.length + b.length];
  for (int i=0; i<a.length; i++)
   c[i] = a[i];
  for (int i=0; i<b.length; i++)
   c[a.length + i] = b[i];
  return c;
}
```

4. (*Esame scritto del 7/11/2003, Esercizio 3.*)
Scrivere un metodo iterativo che, dati una stringa `s` ed un array `a` di interi, restituisce un array di caratteri `c` tale che l'elemento `c[i]` è il carattere di `s` in posizione `a[i]`. Ad esempio, dati `s = "abcd"` e `a = {2,1,0,1,3}`, il metodo restituisce `c = {'c','b','a','b','d'}`. (Nota: si assuma che valga $0 \leq a[i] \leq s.length()-1$ per ogni i.)

```
public static char[] seleziona (String s, int[] a) {
  char[] c = new char[a.length];
  for (int i=0; i<c.length; i++) {
    c[i] = s.charAt(a[i]);
  }
  return c;
}
```

Supponendo che il metodo `seleziona` sia stato definito in una classe `A`, un semplice metodo `main` per provare `seleziona` sull'esempio dato è il seguente:

```
public static void main (String[] args) {
  String s = "abcd";
  int[] a = {2,1,0,1,3};
  char[] c = A.seleziona(s,a);

  for (int i=0; i<c.length; i++)        // stampa dell'array c
    System.out.print(c[i]+" ");
  System.out.println();
}
```

5. (*Esame scritto del 18/9/2002, Esercizio 2.*)
Scrivere un metodo in linguaggio Java che, data una stringa `s` ed un array di caratteri `c`, restituisca un array di interi `conta` tale che `conta[i]` sia il numero di occorrenze del carattere `c[i]` nella stringa `s`. (Esempio: se `s = "ciao pippo"` e `c = {'a', 'b', 'c', 'p'}`, il metodo restituisce `conta = {1,0,1,3}`).

```
public static int[] ContaChar (String s, char[] c) {
  int[] conta = new int[c.length];
  for (int i = 0; i < c.length; i++) {
    for (int j = 0; j < s.length(); j++) {
      if (s.charAt[j] == c[i])
         conta[i]++;
    }
  }
  return conta;
}
```

6. (*Esame scritto del 24/3/2004, Esercizio 3.*)
Scrivere un metodo che, dati un array a di stringhe ed un intero k, restituisce l'array a ruotato di k posizioni verso destra.
Ad esempio, dati `a ={"winnie","pimpi","ro", "tigro","kanga"}` e `k=3`, il metodo restituisce l'array `{"ro","tigro","kanga","winnie", "pimpi"}`.

```
public static String[] rotazione (String[] a, int k) {
  int n = a.length;
  String[] b = new String[n];
  for (int i=0; i<n; i++) {
    b[(i+k)%n] = a[i];
  }
  return b;
}
```

7. (*Esame scritto del 18/7/2003, Esercizio 3*)
   Scrivere un metodo che, data una stringa `s`, restituisce un array di stringhe contenente tutti i prefissi di `s` in *ordine crescente* (inclusa `s`). Ad esempio, data la stringa `"abcdef"`, il metodo restituisce l'array {`"a","ab","abc","abcd","abcde","abcdef"`}.

```
public static String[] prefissi (String s) {
  int n = s.length();
  String[] a = new String[n];
  for (int i=0; i<n; i++)
    a[i] = s.substring(0,i+1);
  return a;
}
```

8. (*Esame scritto del 13/12/2002, Esercizio 2.*)
   Scrivere un metodo che, dati tre interi `i`, `j` ed `n`, restituisce un array di interi di lunghezza `n`, i cui primi due elementi sono `i` e `j`, ed ogni elemento successivo è uguale al prodotto dei due elementi che lo precedono nell'array. Ad esempio, dati `i=3`, `j=-2` ed `n=5`, il metodo restituisce l'array {`3,-2,-6,12,-72`}. (Si assuma `n` $\geq$ 2.)

```
public static int[] metodo (int i, int j, int n) {
  int[] a = new int[n];
  a[0] = i;
  a[1] = j;
  for (int k=2; k<n; k++) {
    a[k] = a[k-2] * a[k-1];
  }
  return a;
}
```

   Supponendo che `metodo` sia definito nella classe `ArrayProp`, un semplice metodo `main` per provare `metodo` è il seguente:

```
public static void main (String[] args) {
  int i = Integer.parseInt(args[0]);
  int j = Integer.parseInt(args[1]);
  int n = Integer.parseInt(args[2]);
  int[] b = ArrayProp.metodo(i,j,n);
  for (int k=0; k<b.length; k++) {
     System.out.print(b[k] + " ");
  }
  System.out.println();
}
```

Altri esercizi

9. Scrivere un metodo che, dato un array bidimensionale di interi a, restituisce *true* se tutte le righe di a contengono almeno un numero negativo, altrimenti il metodo restituisce *false*.

```
public static boolean negArrayBi (int[][] a) {
  int i = 0;
  while (i < a.length) {
    int j = 0;
    boolean trovato = false;
    while (j < a[i].length && !trovato) {
      if (a[i][j] < 0)
        trovato = true;
      else
        j++;
    }
    if (!trovato)
      return false;
    i++;
  }
  return true;
}
```

10. (*Esame scritto del 4/11/2002, soluzione iterativa per una versione dell'Esercizio 4*.)
Scrivere un metodo *iterativo* che, dato un array `a` di stringhe, restituisce `true` se `a` è ordinato in modo crescente rispetto alla lunghezza dei suoi elementi, e `false` altrimenti. Ad esempio, dato `a = {"ad", "cbah", "wqzxtr"}`, il metodo restituisce `true`.

```
public static boolean crescente (String[] a) {
  for (int i=0; i<a.length-1; i++) {
     if (a[i].length() >= a[i+1].length())
       return false;
  }
  return true;
}
```

11. (*Esame scritto del 15/9/2004, Esercizio 3*.)
Scrivere un metodo che, dati un array a di stringhe ed un intero k ($k \geq 2$), restituisce *true* se in a esistono almeno k stringhe di uguale lunghezza, altrimenti il metodo restituisce *false*. Ad esempio, dato `a = {"leone","cane","gatto","pesce","elefante"}` e `k=3`, il metodo restituisce *true*.

```
public static boolean m (String[] a, int k) {
  int i = 0;
  int cont = 1;
  while (i < a.length) {
    int j = i+1;
    while (j < a.length) {
      if (a[i].length() == a[j].length()) {
        cont++;
        if (cont >= k) return true;
```

```
      }
      j++;
    }
    i++;
    cont = 1;
  }
  return false;
}
```

12. (*Esame scritto del 14/7/2004, Esercizio 3.*)
Scrivere un metodo che, dati due array a e b di interi, restituisce *true* se tutti gli elementi dell'array b compaiono nell'array a nello stesso ordine in cui compaiono in b, altrimenti il metodo restituisce *false*. Ad esempio, dati gli array `a = {-5,4,7,-1,10,21,9,-7}` e `b = {4,-1,9,-7}`, il metodo restituisce *true*.

```
public static boolean ordineArray (int[] a, int[] b) {
  int i = 0, j = 0;
  while (j < b.length && i < a.length) {
    if (b[j] == a[i]) {
      i++;
      j++;
    }
    else i++;
  }
  if (j == b.length) return true;
  else return false;
}
```

13. (*Esame scritto del 4/11/2002, una versione dell'Esercizio 1.*)
Si consideri il seguente frammento di codice in Java:

```
int x = 1;
for (int j = 1; j <= n; j++)
 x = x+j;
System.out.print(x);
```

Considerando le tre varianti del frammento ottenute sostituendo alcuni degli identificatori come definito nei seguenti casi, individuare eventuali errori o dire quale valore viene stampato:
1) `x` sostituito da `i` ed `n` sostituito da 5;
2) `x` sostituito da `j` ed `n` sostituito da 10;
3) `x` sostituito da `n`.

1) 16
2) errore in compilazione (`j` già definita)
3) -2147450879 (il valore di `n` cresce piú velocemente del valore di `j`, per cui il comando `for` sarebbe eseguito all'infinito se non fosse che ad un certo punto viene superato il valore positivo massimo di tipo `int`, per cui sommando ulteriormente ad `n` si ottiene un valore negativo per `n`, che fa terminare il `for`).