LABORATORIO DI PROGRAMMAZIONE 2002-2003

Prova Intermedia - 4 Novembre 2002

Soluzione degli Esercizi

Esercizio 1.

v1. 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`.

Soluzione
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`).

v2. Si consideri il seguente frammento di codice in Java:

```
int y = 1;
for (int j = 1; j <= m; j++)
 y = y+j;
System.out.print(y);
```

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) `y` sostituito da `j` ed `m` sostituito da 9;
2) `y` sostituito da `i` ed `m` sostituito da 6;
3) `y` sostituito da `m`.

Soluzione
1) errore in compilazione (`j` già definita)
2) 22
3) -2147450879

v3. Si consideri il seguente frammento di codice in Java:

```
int z = 1;
for (int k = 1; k <= n; k++)
 z = z*k;
System.out.print(z);
```

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) `z` sostituito da `i` ed `n` sostituito da 4;
2) `z` sostituito da `k` ed `n` sostituito da 8;
3) `z` sostituito da `n`.

Soluzione
1) 24
2) errore in compilazione (`k` già definita)
3) 1

v4. Si consideri il seguente frammento di codice in Java:

```
int w = 1;
for (int j = 1; j <= m; j++)
 w = w*j;
System.out.print(w);
```

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) `w` sostituito da `j` ed `m` sostituito da 6;
2) `w` sostituito da `i` ed `m` sostituito da 5;
3) `w` sostituito da `m`.

Soluzione
1) errore in compilazione (`j` già definita)
2) 120
3) 1

Esercizio 2.

v1. Si consideri il seguente frammento di codice in Java:

```
int[][] a = b;
int i=a.length-1, j=a[0].length-1;
int cont = 0;
while (i >= 0) {
 while (j > 0) {
  if (a[i][j] == a[i][j-1]) cont++;
  j--;
 }
 i--;
 j = a[0].length-1;
}
```

Determinare il valore finale delle variabili `i`, `j` e `cont` nei seguenti casi:
1) `b = {{3,5,5},{4,4,3}};`
2) `b = {{7,7},{7,3},{3,9}}`.

Soluzione
1) `i = -1`, `j = 2`, `cont = 2`.
2) `i = -1`, `j = 1`, `cont = 1`.

v2. Si consideri il seguente frammento di codice in Java:

```
int[][] a = b;
int i=a.length-1, j=a[0].length-1;
int cont = 0;
while (j >= 0) {
 while (i > 0) {
  if (a[i][j] == a[i-1][j]) cont++;
  i--;
 }
 j--;
 i = a.length-1;
}
```

Determinare il valore finale delle variabili `i`, `j` e `cont` nei seguenti casi:
1) `b = {{4,5,-3},{4,4,3}};`
2) `b = {{7,7},{7,3},{3,9}}`.

Soluzione
1) `i = 1`, `j = -1`, `cont = 1`.
2) `i = 2`, `j = -1`, `cont = 1`.

v3. Si consideri il seguente frammento di codice in Java:

```
int[][] a = b;
int i = a.length-1, j;
int cont = 0;
while (i >= 0) {
 j = a[0].length-1;
 while (j > 0) {
  if (a[i][j] != a[i][j-1]) cont++;
  j--;
 }
 i--;
}
```

Determinare il valore finale delle variabili `i`, `j` e `cont` nei seguenti casi:
1) `b = {{3,5,5},{4,4,3}};`
2) `b = {{7,7},{7,3},{3,9}}`.

Soluzione
1) `i = -1`, `j = 0`, `cont = 2`.
2) `i = -1`, `j = 0`, `cont = 2`.

v4. Si consideri il seguente frammento di codice in Java:

```
int[][] a = b;
int i, j = a[0].length-1;
int cont = 0;
while (j >= 0) {
 i = a.length-1;
 while (i > 0) {
  if (a[i][j] != a[i-1][j]) cont++;
  i--;
 }
 j--;
}
```

Determinare il valore finale delle variabili `i`, `j` e `cont` nei seguenti casi:
1) `b = {{4,5,3},{4,4,3}};`
2) `b = {{7,7},{7,3},{9,3}}`.

Soluzione
1) `i = 0`, `j = -1`, `cont = 1`.
2) `i = 0`, `j = -1`, `cont = 2`.

Esercizio 3.

v1. 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}`.

Soluzione

```
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;
}
```

v2. 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}`.

Soluzione

```
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;
}
```

v3. Scrivere un metodo che, dato un array `a` di numeri in virgola mobile, restituisce l'array con gli elementi di `a` rovesciati. Ad esempio, dato `a = {-7.2,3.1,12.5,1.9}`, il metodo restituisce l'array `{1.9,12.5,3.1,-7.2}`.

Soluzione

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

v4. Scrivere un metodo che, dati due array `a` e `b` di numeri in virgola mobile, restituisce l'array in cui `b` è stato attaccato alla fine di `a`. Ad esempio, dati `a = {7.1,-3.3}` e `b = {3.2,4.4,1.5,-1.3}`, il metodo restituisce l'array `{7.1,-3.3,3.2,4.4,1.5,-1.3}`.

Soluzione

```
public static double[] append (double[] a, double[] b) {
  double[] c = new double[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;
}
```

Esercizio 4.

v1. Scrivere un metodo ricorsivo 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`.

Soluzione

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

v2. Scrivere un metodo ricorsivo che, dato un array `a` di stringhe, restituisce `true` se `a` è ordinato in modo decrescente rispetto alla lunghezza dei suoi elementi, e `false` altrimenti. Ad esempio, dato `a = {"welcome", "ciao", "hi!"}`, il metodo restituisce `true`.

Soluzione

```
public static boolean decrescente (String[] a) {
  return decrescenteRic (a,0);
}
public static boolean decrescenteRic (String[] a, int i) {
  if (i == a.length-1)
    return true;
  if (a[i].length() > a[i+1].length())
    return decrescenteRic (a,i+1);
  else
    return false;
}
```

v3. Scrivere un metodo ricorsivo che, dato un array `a` di stringhe, restituisce `true` se `a` è ordinato in modo non crescente rispetto alla lunghezza dei suoi elementi, e `false` altrimenti. Ad esempio, dato `a = {"oops", "ops", "acc"}`, il metodo restituisce `true`.

Soluzione

```
public static boolean noncrescente (String[] a) {
  return noncrescenteRic (a,0);
```

```
}
public static boolean noncrescenteRic (String[] a, int i) {
  if (i == a.length-1)
    return true;
  if (a[i].length() >= a[i+1].length())
    return noncrescenteRic (a,i+1);
  else
    return false;
}
```

v4. Scrivere un metodo ricorsivo che, dato un array `a` di stringhe, restituisce `true` se `a` è ordinato in modo non decrescente rispetto alla lunghezza dei suoi elementi, e `false` altrimenti. Ad esempio, dato `a = {"mosca", "zanzara", "cavallo"}`, il metodo restituisce `true`.

Soluzione

```
public static boolean nondecrescente (String[] a) {
  return nondecrescenteRic (a,0);
}
public static boolean nondecrescenteRic (String[] a, int i) {
  if (i == a.length-1)
    return true;
  if (a[i].length() <= a[i+1].length())
    return nondecrescenteRic (a,i+1);
  else
    return false;
}
```

Esercizio 5.

v1. Scrivere un metodo che, dato un array bidimensionale quadrato `a` di caratteri, restituisce la stringa ottenuta concatenando i caratteri presenti nel triangolo in alto a sinistra inclusa la diagonale. Ad esempio, data la matrice `a`

```
{{'a','b','c'},
 {'d','e','f'},
 {'g','h','i'}}
```

il metodo restituisce la stringa `"abcdeg"`.

Soluzione

```
public static String triangolo (char[][] a) {
  int m = a.length;
  String s = "";
  for (int i=0; i<m; i++)
    for (int j=0; j<m-i; j++)
      s = s+a[i][j];
  return s;
}
```

v2. Scrivere un metodo che, dato un array bidimensionale quadrato `a` di caratteri, restituisce la stringa ottenuta concatenando i caratteri presenti nel triangolo in alto a destra inclusa la diagonale. Ad esempio, data la matrice `a`

```
    {{'a','b','c'},
     {'d','e','f'},
     {'g','h','i'}}
```

il metodo restituisce la stringa `"abcefi"`.

Soluzione

```
public static String triangolo (char[][] a) {
  int m = a.length;
  String s = "";
  for (int i=0; i<m; i++)
    for (int j=i; j<m; j++)
      s = s+a[i][j];
  return s;
}
```

v3. Scrivere un metodo che, dato un array bidimensionale quadrato `a` di caratteri, restituisce la stringa ottenuta concatenando i caratteri presenti nel triangolo in basso a sinistra inclusa la diagonale. Ad esempio, data la matrice `a`

```
    {{'a','b','c'},
     {'d','e','f'},
     {'g','h','i'}}
```

il metodo restituisce la stringa `"adeghi"`.

Soluzione

```
public static String triangolo (char[][] a) {
  int m = a.length;
  String s = "";
  for (int i=0; i<m; i++)
    for (int j=0; j<=i; j++)
      s = s+a[i][j];
  return s;
}
```

v4. Scrivere un metodo che, dato un array bidimensionale quadrato `a` di caratteri, restituisce la stringa ottenuta concatenando i caratteri presenti nel triangolo in basso a destra inclusa la diagonale. Ad esempio, data la matrice `a`

```
    {{'a','b','c'},
     {'d','e','f'},
     {'g','h','i'}}
```

il metodo restituisce la stringa `"cefghi"`.

Soluzione

```
public static String triangolo (char[][] a) {
  int m = a.length;
  String s = "";
  for (int i=0; i<m; i++)
    for (int j=m-i-1; j<m; j++)
      s = s+a[i][j];
  return s;
}
```

Esercizio 6.

v1-v3. Scrivere un metodo ricorsivo che, date due stringhe non vuote `s` e `t`, restituisce la posizione del primo carattere della prima occorrenza di `t` in `s` se `t` è sottostringa di `s`, mentre restituisce -1 se `t` non è sottostringa di `s`. Ad esempio, date `s = "albcdebcw"` e `t = "bc"`, il metodo restituisce 2, mentre se `s = "wrtyvfdyvft"` e `t = "yvf"`, il metodo restituisce 3.

Soluzione

```
public static int pos_substring (String s, String t) {
   return pos_substringRic (s,t,0);
}
public static int pos_substringRic (String s, String t, int i) {
  if (s.length()-t.length() < i)
    return -1;
  if (t.equals(s.substring(i,i+t.length())))
      return i;
  else
      return pos_substringRic (s,t,i+1);
}
```

v2-v4. Scrivere un metodo ricorsivo che, date due stringhe non vuote `s` e `t`, restituisce `true` se `t` è sottostringa di `s`, e `false` altrimenti. Ad esempio, date `s = "albcdebcw"` e `t = "cdeb"`, il metodo restituisce `true`, e se `s = "persepolis"` e `t = "poli"`, il metodo restituisce `true`.

Soluzione

```
public static boolean is_substring (String s, String t) {
   return is_substringRic (s,t,0);
}
public static boolean is_substringRic (String s, String t, int i) {
  if (s.length()-t.length() < i)
    return false;
  if (t.equals(s.substring(i,i+t.length())))
      return true;
  else
      return is_substringRic (s,t,i+1);
}
```