Prova scritta dell'esame di Fondamenti di Informatica – 25 Marzo 2002 – Soluzione della Traccia B

Esercizio 1

Il metodo verifica che ogni elemento dell'array `y` sia divisibile per l'elemento dell'array `x` che si trova nella posizione opposta. Se i due array hanno dimensioni diverse il metodo restituisce `null`, altrimenti restituisce un array di booleani di dimensioni pari agli array in input, nel quale l'elemento in posizione `i` ha valore `true` se l'elemento dell'array `y` di indice `i` è divisibile per l'elemento opposto (ovvero, di indice `x.length-1-i`) di `x`, e `false` altrimenti. Nel caso in cui il valore dei parametri sia `x=[6,3,4,7]` e `y=[14,12,6,12]` il contenuto dell'array `n` restituito sarà `[true,true,true,true]`, in quanto `14%7==0`, `12%4==0`, `6%3==0` e `12%6==0`.

Esercizio 2

```
public static void testprefisso (String s)
{
  int posizione = s.indexOf('#');
  if (posizione == -1)
    System.out.println ("nessun suffisso");
  else
  {
    String inizio = s.substring(0,posizione);
    String fine = s.substring(s.length()-inizio.length());
    if (inizio.equals(fine))
      System.out.println ("suffisso corretto");
    else
      System.out.println ("suffisso scorretto");
  }
}
```

Esercizio 3

```
public class Esercizio3
{
  private static double[][] estrai (double A[][], int r, int c, int dim)
  {

    double R[][] = new double[dim][dim];
    for (int k = r; k < r+dim; k++)
      for (int h = c; h < c+dim; h++)
        R[k-r][h-c] = A[k][h];
    return R;
  }

  private static double[][] prodotto (double A[][], double B[][])
  {
    double C[][] = new double[A.length][B[0].length];
    for (int i=0; i<A.length; i++)
      for (int j=0; j<B[0].length; j++)
      {
        C[i][j]=0;
        for (int k=0; k<A[0].length; k++)
          C[i][j]+=A[i][k]*B[k][j];
      }
    return C;
  }



private static boolean righePariOrdinate (double A[][])
  {
    boolean ordinate = true;
    for (int i = 0; i < A.length && ordinate; i=i+2)
      for (int j = 0; j < A[0].length-1 && ordinate; j++)
        if (A[i][j] >= A[i][j+1])
          ordinate = false;
```

```
    return ordinate;
  }

  private static double[][] differenza (double A[][], double B[][])
  {
    double[][] C = new double[A.length][A[0].length];
    for (int i = 0; i < A.length; i++)
      for (int j = 0; j < A[0].length; j++)
        C[i][j] = A[i][j] - B[i][j];
    return C;
  }

  private static void leggi (double M[][])
  {
    for (int i = 0; i < M.length; i++)
      for (int j = 0; j < M[0].length; j++)
        M[i][j] = Console.readDouble ("Elemento in posizione "+i+","+j+" =");
  }

  private static void stampa (double M[][])
  {
    for (int i = 0; i < M.length; i++)
    {
      for (int j = 0; j < M[0].length; j++)
        System.out.print (M[i][j]+"\t");
      System.out.println();
    }
  }

  public static void main (String args[])
  {
    int dim = 10;
    double A[][] = new double [dim][dim];

    System.out.println ("Lettura della matrice A:");
    leggi(A);

    double A11[][] = estrai (A,0,0,dim/2);
    double A12[][] = estrai (A,0,dim/2,dim/2);
    double A21[][] = estrai (A,dim/2,0,dim/2);
    double A22[][] = estrai (A,dim/2,dim/2,dim/2);

    double B[][] = differenza (prodotto(A11,A21),prodotto(A12,A22));

    System.out.println ("Stampa della matrice B:");
    stampa (B);

    if (righePariOrdinate (B))
      System.out.println("Gli elementi delle righe di indice "+
                         "pari sono ordinati in modo crescente");
    else
      System.out.println("Gli elementi delle righe di indice "+
                         "pari NON sono tutti ordinati in modo crescente");
  }
}
```

Esercizio 4

```
public static boolean verificaprodotto (int []V1, int []V2)
 {
  int prodPari = 1;
  for (int i = 0; i < V1.length; i+=2)
    prodPari *= V1[i];
  int prodDispari = 1;
  for (int i = 1; i < V2.length; i+=2)
    prodDispari *= V2[i];
  return (prodPari == prodDispari);
}
```