Esercizio 3. Scrivere un metodo che, date due stringhe $s1$ ed $s2$ ed un intero k (k>0), restituisce *true* se nella stringa $s1$ esistono *almeno* k sottostringhe uguali ad $s2$, altrimenti il metodo restituisce *false*. Ad esempio, dati $s1$ = `"aabbbcbbadvbbc"`, $s2$ = `"bb"` e k=4, il metodo restituisce *true*.

```
public static boolean substringSearch (String s1, String s2, int k) {
  if (k <= 0)
     return false;
  int l2 = s2.length(), c = 0;
  for (int i=0; i <= s1.length()-l2; i++) {
    if (s2.equals(s1.substring(i,i+l2))) {
      c++;
      if (c >= k) return true;
    }
  }
  return false;
}
```

Esercizio 4. Scrivere un metodo che, dati due array monodimensionali di interi a e b ordinati in senso non crescente (ovvero, $a[0] \geq a[1] \geq a[2] \ldots$ e $b[0] \geq b[1] \geq b[2] \ldots$), restituisce un array monodimensionale di interi c che contiene gli elementi di a e b (eventualmente con ripetizioni di elementi) ordinati in senso non crescente.
Ad esempio, se a = {5,2,-1,-4,-10,-21} e b = {7,2,-3,-4,-7}, il metodo restituisce l'array c = {7,5,2,2,-1,-3,-4,-4,-7,-10,-21}.

```
public static int[] mergeNC (int[] a, int[] b) {
  int[] c = new int[a.length + b.length];
  int i = 0, j = 0, k = 0;
  while (i < a.length && j < b.length) {
    if (a[i] > b[j]) {
      c[k] = a[i];
      i++;
    }
    else {
      c[k] = b[j];
      j++;
    }
    k++;
  }

  if (i >= a.length) {
     while (j < b.length) {
       c[k] = b[j];
       k++;
       j++;
     }
  }
```

```
  else {
    while (i < a.length) {
      c[k] = a[i];
      k++;
      i++;
    }
  }
  return c;
}
```

Esercizio 5. Scrivere un metodo che, dato un array bidimensionale di stringhe a, restituisce un array monodimensionale di stringhe b tale che l'elemento $b[i]$ è la prima stringa nell'array $a[i]$ di lunghezza massima in $a[i]$. Ad esempio, dato $a = \{\{$`"ad","c","abc","ab","bbc"`$\}$, $\{$`"ckk","bc","rew","mghgh"`$\}$, $\{$`"argh","xxc","beta"`$\}\}$, il metodo restituisce l'array b dato da $\{$`"abc","mghgh","argh"`$\}$.
(Si assuma che ogni riga $a[i]$ contenga almeno una stringa.)

```
public static String[] maxLenFirst (String[][] a) {
  String[] b = new String[a.length];

  for (int i=0; i<a.length; i++) {
    String s = a[i][0];
    for (int j=1; j<a[i].length; j++) {
      if (a[i][j].length() > s.length()) {
        s = a[i][j];
      }
    }
    b[i] = s;
  }

  return b;
}
```