Prova d'Esame del 22/07/2016

Esercizio 1. Si implementi in Java un metodo che riceva come parametro una matrice di caratteri (sia m1) e restituisca una matrice di interi (sia m2), secondo il funzionamento descritto di seguito.
La matrice può contenere qualunque carattere; tra questi, un significato particolare è ricoperto dai caratteri "|" (pipe) e "-" (dash): sono caratteri "infettanti". In particolare, se in una posizione compare il carattere "|", questo "infetta" tutti i caratteri che compaiono nella stessa colonna; analogamente, il carattere "-" infetta tutti i caratteri che compaiono sulla stessa riga. Si noti che un carattere può essere infettato tanto "orizzontalmente" (dal carattere "-") quanto verticalmente (dal carattere "|"), e può pertanto essere infettato più volte (anche più volte orizzontalmente e/o verticalmente). La matrice da restituire m2 dovrà essere creata delle stesse dimensioni di m1; ogni posizione della matrice restituita m2 dovrà contenere un numero intero pari al numero di "infezioni" subite (quindi pari al numero di caratteri "infettanti" che lo interessano). Una posizione conterrà pertanto il numero "0" se il carattere nella corrispondente posizione di m2 è "sano", cioè non "infettato" da alcun altro carattere. Nelle posizioni di m2 corrispondenti ai caratteri "infettanti" in m1 dovrà comparire il numero "-1".
ESEMPIO: Di seguito è riportato un esempio di matrice m1 in input (a sinistra) e la corrispondente matrice di output m2 da restituire (a destra).

a	r	g	z	c	o		0	0	0	1	0	1
b	a	u	|	v	u		0	0	0	-1	0	1
r	f	y	t	b	t		0	0	0	1	0	0
-	a	-	f	g	r		-1	2	-1	3	2	2
s	d	n	d	u	e		0	0	0	1	0	1
z	-	d	h	t	|		1	-1	1	2	1	-1
			m1							m2		

SOLUZIONE:

```
public int[][] esercizio1(char[][] m1){
      //dichiamo la matrice m2 delle stesse dimensioni di m1
      int[][] m2 = new int[m1.lenght][m1[0].lenght];
      for(int i=0;i<m2.length;i++){
            for(int j=0;j<m2[0].length;j++){
                  //se e' un carattere infettivo, assegno -1
                  if(m1[i][j]=='|' || m1[i][j]=='-')
                        m2[i][j]=-1;
                  else{
                        //altrimenti conto quanti caratteri infettivi
                        //ci sono in riga i e colonna j
                        m2[i][j]=contaInfetteRiga(i,m1) +
                                          contaInfetteColonna(j,m1);
                  }
            }
      }
return m2;
}

public int contaInfetteRiga(int riga, char[][] m){
      int cont=0;
      for(int i=0;i<m[riga].length;i++){
      //conta il numero di caratteri '-' infettivi
            if(m[riga][i]=='-')
                  cont++;
      }
```


CAMPUS DI ARCAVACATA - Ponte Pietro Bucci Cubo 30B - 87036 Arcavacata di Rende (CS) tel. (+39) 0984 496400 fax (+39) 0984 496410
http://sv.mat.unical.it

Prova d'Esame del 22/07/2016

```
            return cont;
      }

      public int contaInfetteColonna(int colonna, char[][] m){
            int cont=0;
            for(int i=0;i<m.length;i++){
            //conta il numero di caratteri '|' infettivi
                  if(m[i][colonna]=='|' )
                        cont++;
            }
            return cont;
      }
```

Esercizio 2. Si implementi in Java un metodo "booleano" (che restituisca, quindi, "true" o "false") che riceva in input un array di interi, sia "v". Il metodo deve restituire "true" se "v" è di fatto composto da due parti ("sotto-array") perfettamente identiche e consecutive; deve restituire "false", altrimenti. È obbligatorio risolvere il problema SIA con un approccio "iterativo" che con un approccio "ricorsivo". *ESEMPIO:* Se l'array "v" fosse uno dei seguenti, allora il metodo dovrebbe restituire "true" (in entrambe le versioni, iterativa e ricorsiva, ovviamente).

1	2	3	14	1	2	3	14

1	7	5	3	4	1	7	5	3	4

8	5	2	9	9	2	8	5	2	9	9	2

SOLUZIONE:

```
boolean duplicatoIter(int[] v){
//se la dimensione e' zero o dispari ritorna false
if((v.length == 0) || (v.lenght%2 != 0))
      return false;
int dim = v.length/2; //dimensione dei sotto array
for(int i=0;i<dim;i++){
//controllo gli elementi rispettivi nei due sottoarray
      if(v[i] != v[dim+i])
      return false; //appena trovo uno diverso mi fermo
}
return true;
}

boolean duplicatoRic(int[] v){
//se la dimensione e' zero o dispari ritorna falso
if((v.length == 0) || (v.lenght%2 != 0))
      return false;
int dim = v.length/2;
return ugualiRicorsivo(v,0,dim);
}

boolean ugualiRicorsivo(int[] v, int posizioneCorrente, int dim){
//l'array v e' stato visto tutto, posso restituire true
if(posizioneCorrente>=dim) return true;
//verifico che i due rispettivi elementi dei due sottoarray non siano diversi
if(v[posizioneCorrente]!=v[posizioneCorrente+dim])
      return false;
//continuo ricorsivamente sul prossimo elemento dell'array
return ugualiRicorsivo(v, posizioneCorrente+1,dim);
}
```


CAMPUS DI ARCAVACATA - Ponte Pietro Bucci Cubo 30B - 87036 Arcavacata di Rende (CS) tel. (+39) 0984 496400 fax (+39) 0984 496410
http://sv.mat.unical.it

Prova d'Esame del 22/07/2016

Esercizio 3. Si implementi in Java un metodo che legga da input una sequenza di numeri interi positivi terminata da un "tappo" negativo che restituisca il numero di elementi presenti nella sequenza il cui valore corrisponde alla posizione in cui compaiono nella sequenza stessa. *ESEMPIO:* se la sequenza fosse la seguente:

4 8 **3** 7 9 1 **7** 7 3 2 **11** 1 3 4 -1

allora il metodo dovrebbe restituire il numero 3. Infatti, nella sequenza sono presenti 3 numeri (evidenziati in grassetto qui sopra) il cui valore corrisponde alla posizione in cui compaiono: il 3 compare in terza posizione, il 7 in settima posizione, ed infine il numero 11 compare in undicesima posizione.

SOLUZIONE:

```
int posizioneInSequenza(){
    var totale=0,posizioneCorrente=0;
    Scanner s = new Scanner(System.in);
    int num = s.nextInt();
    while(num>=0){
      if(num==posizioneCorrente)
        totale++;
      posizioneCorrente++;
      num = s.nextInt();
    }
    return totale;
}
```


CAMPUS DI ARCAVACATA - Ponte Pietro Bucci Cubo 30B - 87036 Arcavacata di Rende (CS) tel. (+39) 0984 496400 fax (+39) 0984 496410
http://sv.mat.unical.it