Prova d'Esame del 17/02/2014

ESERCIZIO 1. Si implementi in java una classe, chiamata "Esame", che contenga ALMENO 3 metodi statici: un main() e altri due metodi, siano met1() e met2(), che svolgano rispettivamente gli esercizi descritti di seguito. Il candidato è fortemente incoraggiato a progettare e implementare all'interno della stessa classe ulteriori altri metodi ritenuti utili per completare il compito.
Il "main()" dovrà essere implementato in modo da provare opportunamente entrambi gli esercizi.

Esercizio 1A. Si progetti met1() in modo tale che, ricevuti come parametri due array di INTERI, siano v1 e v2, restituisca TRUE se e solo se ogni elemento di v1 che ha (almeno) un multiplo in v2, compare almeno due volte in v1. Il metodo DEVE essere implementato facendo uso di RICORSIONE.
Esempio: se i due array fossero

v1= {3, 5, 6, 5, 3, 5} v2 = {1, 20, 8, 16, 21, 1, 13}

met1() dovrebbe restituire TRUE; infatti, i numeri in v1 che hanno almeno un multiplo in v2 sono 3 e 5, ed entrambi compaiono almeno due volte in v1.
Se invece i due array fossero

v1= {**3**, 5, 6, 5, 5} v2 = {1, 20, 8, 16, 21, 1, 13}

allora met1()dovrebbe restituire FALSE: infatti, in questo caso v1 ha un elemento (il numero 3, evidenziato nell'esempio qui sopra) che vi compare una volta sola, pur avendo un multiplo in v2 (il numero 21).

SOLUZIONE:

```
import java.util.*;

public class Esame {

	public static Scanner input = new Scanner(System.in);

	public static void leggiArray(int a[]){
	   System.out.println("Inserire " + a.length + " numeri interi, elementi di un array");
	   for (int i=0; i < a.length; i++)
	         a[i] = input.nextInt();
	}

	public static boolean esisteMultiplo(int x, int v[]) {
		for (int i = 0; i < v.length; i++)
			if (v[i] % x == 0)
				return true;
		return false;
	}

	public static boolean comparePiuVolte(int x, int v[]) {
		boolean giaPresenteUnaVolta = false;
		for (int i = 0; i < v.length; i++)
			if (v[i] == x)
				if (giaPresenteUnaVolta)
					return true;
				else
					giaPresenteUnaVolta = true;
		return false;
	}

	public static boolean met1(int v1[], int v2[], int i) {
		if (i > v1.length - 1)
			return true;
```


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 17/02/2014

```
            if (esisteMultiplo(v1[i], v2) && !comparePiuVolte(v1[i], v1))
                  return false;
            return met1(v1, v2, i+1);
      }

      public static void main(String[] args) {
            System.out.println(" Quanti elementi conterra' il primo array? ");
            int v1[] = new int[input.nextInt()];
            leggiArray(v1);
            System.out.println(" Quanti elementi conterra' il secondo array? ");
            int v2[] = new int[input.nextInt()];
            leggiArray(v2);
            if ( met1(v1,v2, 0) )
                System.out.println(" OK!! ");
            else
                System.out.println(" NOT OK!! ");
      }
}
```

Esercizio 1B. Si progetti met2() in modo tale che, ricevuti come parametri una matrice di caratteri M, due interi X ed Y ed un array di caratteri V, restituisca TRUE se e solo se, a partire dall'elemento di M in posizione [X,Y], la parola rappresentata da V è presente nella matrice. Si tenga conto di quanto specificato di seguito.

- Si deve cercare la parola, lettera per lettera, partendo dalla cella [X,Y] e passando da una cella all'altra, una alla volta;
- Il passaggio da una cella all'altra, nella matrice M, deve avvenire solo tra celle adiacenti. Consideriamo come adiacenti ad una cella la cella superiore, quella inferiore, quella a sinistra e quella a destra (quindi, in genere, una cella ha 4 celle adiacenti, a meno che non sia sulla cornice esterna della matrice, caso in cui si hanno meno di 4 celle adiacenti).
- Si può assumere che non ci sia la possibilità di più percorsi: o la lettera successiva è presente una volta tra le celle adiacenti alla cella corrente, oppure non è presente affatto.
- È possibile tornare più volte sulla stessa cella; non è importante, e non bisogna curarsi affatto di ricordarsi dove si è già stati.
- Se, passando da una cella all'altra, è possibile trovare tutte le lettere di V nello stesso ordine, allora il metodo dovrà restituire true; altrimenti, dovrà restituire false.

m	r	**l**	**e**	w
e	**m**	**a**	w	u
x	**i**	**n**	p	m
v	d	w	e	h

Esempio: ricevuti come parametri la matrice qui a lato, gli interi 1 e 2 e l'array di caratteri V= {'a', 'n', 'i', 'm', 'a', 'l', 'e'} il metodo "met2() dovrebbe restituire "true".

SOLUZIONE:

```
import java.util.*;

public class Esame {

      public static Scanner input = new Scanner(System.in);

      public static void leggiCharMatrice(char m[][]) {
            System.out.println("Inserire gli elementi della matrice - " + m.length + " righe, ");
            for (int i = 0; i < m.length; i++) {
                  System.out.println(m[i].length + " elementi per la riga " + i);
                  for (int j = 0; j < m[i].length; j++)
```


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 17/02/2014

```
                        m[i][j] = input.next().charAt(0);
                System.out.println();
        }
    }

    public static void leggiCharArray(char a[]) {
        System.out.println("Inserire " + a.length + " caratteri, elementi di un array");
        for (int i = 0; i < a.length; i++)
            a[i] = input.next().charAt(0);
    }

    public static boolean adiacenteOk(char m[][], int x, int y, char c) {
        if ( x >= 0 && x < m.length && y >= 0 && y < m[0].length && m[x][y] == c )
            return true;
        return false;
    }

    public static boolean met2(char m[][], int x, int y, char v[]) {
        if ( v[0] != m[x][y] )
            return false;
        for (int i = 1; i < v.length; i++) {
            if ( adiacenteOk(m,x-1,y,v[i]) )
                x= x-1;
            else if ( adiacenteOk(m,x+1,y,v[i]) )
                x = x+1;
            else if ( adiacenteOk(m,x,y-1,v[i]) )
                y = y-1;
            else if ( adiacenteOk(m,x,y+1,v[i]) )
                y = y+1;
            else
                return false;
        }
        return true;
    }

    public static void main(String[] args) {
        System.out.println("Inserire le dimensioni della matrice");
        System.out.print("Numero di righe   : ");
        int nr = input.nextInt();
        System.out.print("Numero di colonne : ");
        int nc = input.nextInt();
        char M[][] = new char[nr][nc];
        leggiCharMatrice(M);
        System.out.println("Inserire l'indice di riga da cui partire");
        int X = input.nextInt();
        System.out.println("Inserire l'indice di colonna da cui partire");
        int Y = input.nextInt();
        System.out.println("Inserire la dimensione dell'array");
        char V[] = new char[input.nextInt()];
        leggiCharArray(V);
        if (met2(M, X, Y, V))
            System.out.println(" OK!! ");
        else
            System.out.println(" NOT OK!! ");
    }
}
```


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