UNIVERSITÀ DELLA CALABRIA

## Traccia prova d'esame

La startup Solaire ha creato un nuovo pannello solare che promette enormi passi in avanti nella conservazione e nella cattura dell'energia solare. Un pannello di questo tipo è rappresentato da una matrice `n` x `n` ed è diviso in `s` settori; ogni settore è rappresentato da una matrice `m` x `m` (uguale dimensione per ogni settore) e ogni cella contiene un numero, rappresentante il settore di appartenenza.

Per permettere un funzionamento efficiente, si devono inserire sul pannello un numero finito `t` = `k` x `s` (dove `k` è un numero intero > 0) di transistor tenendo conto dei seguenti vincoli:

- ogni riga del pannello deve avere al più `k` transistor,
- ogni colonna del pannello deve avere al più `k` transistor,
- ogni settore deve contenere esattamente `k` transistor.

La Solaire ci ha chiesto di scrivere un programma che, presi in input la matrice rappresentante il pannello con i vari settori e il numero `k`, restituisca la disposizione dei transistor sul pannello, segnando la loro posizione con il simbolo `#`. Se non esiste una disposizione valida, ovvero una disposizione che soddisfi tutti i vincoli, si deve restituire la stringa `IMPOSSIBILE`.

### Struttura dell'input

L'input consiste in un certo numero di righe. La prima riga è nel formato `n s m k`, ovvero quattro numeri interi rappresentanti, rispettivamente, la dimensione `n` del pannello, il numero di settori `s`, la dimensione `m` di ogni settore e il numero `k`. Le successive `n` righe sono nel formato $c_1$ $c_2$ … $c_n$ dove ogni $c_i$ è un numero intero appartenente all'insieme `{0, 1, …, s-1}`. I settori sono `s` e sono identificati dai numeri `0, 1, …, s-1`.

### Struttura dell'output

L'output della soluzione è rappresentato, nel caso in cui esista una disposizione valida, dalla matrice letta in input con l'aggiunta dei simboli `#` contrassegnanti le celle in cui sono stati inseriti i transistor. Altrimenti, l'output è la stringa `IMPOSSIBILE`.

### Esempio input – output

```
6 4 3 1
0 0 0 1 1 1
0 0 0 1 1 1
0 0 0 1 1 1
2 2 2 3 3 3
2 2 2 3 3 3
2 2 2 3 3 3
```

```
# 0 0 1 1 1
0 0 0 # 1 1
0 0 0 1 1 1
2 # 2 3 3 3
2 2 2 3 # 3
2 2 2 3 3 3
```

```
4 4 2 2
0 0 1 1
0 0 1 1
2 2 3 3
2 2 3 3
```

```
# # 1 1
0 0 # #
# # 3 3
2 2 # #
```

Algoritmi e Strutture Dati
Prova d'esame – 04/05/2018

UNIVERSITÀ DELLA CALABRIA

## Regole e istruzioni

- Si può scegliere se usare C++ o Java; in entrambi i casi, si presuppone che lo studente sappia compilare il codice sorgente e avviare l'eseguibile ottenuto tramite terminale/prompt dei comandi.
- Si può assumere che l'input sia sempre corretto.
- Il vostro programma deve **leggere da stdin** e **scrivere su stdout**. La lettura da file o l'hardcoding di un input nel vostro programma **non rappresenta una soluzione corretta**.
- Per effettuare le (vostre) varie prove, potete creare dei file testuali contenenti input di prova e
  - Scrivere o copiare riga per riga il vostro input al programma,
  - Reindirizzare il contenuto del file di input al vostro programma (consigliato).
- Il vostro programma verrà valutato su vari input utilizzando il secondo metodo.

## Come posso reindirizzare su stdin?

- Supponendo di utilizzare C++ e di essere su Linux/OS X

  ```
  cat input.txt | ./programma
  ```

  dove input.txt è un file testuale contenente un input e programma è l'eseguibile ottenuto dalla compilazione del vostro codice sorgente (input.txt e programma devono essere nella stessa cartella)

- Supponendo di utilizzare C++ e di essere su Windows

  ```
  type input.txt | programma.exe
  ```

  dove input.txt è un file contenete un input e programma.exe è l'eseguibile ottenuto dalla compilazione del vostro codice sorgente (input.txt e programma.exe devono essere nella stessa cartella).