AE_01_matrix_reverse

Esercizio 3 - Flip di una immagine in formato lineare

E' fornito uno stream di dati che può essere pensato come un'immagine potenzialmente infinita. Sappiamo che ogni pixel **P** dell'immagine è codificato in formato `RGBA` (Red, Green, Blue, Alpha) e ogni valore di colore R,G,B,A è codificato su 31bit di profondità.

I valori sono quindi codificati su `[0,2^31-1]`.

Sappiamo a priori quanto è lunga ciascuna riga, cioè conosciamo il numero di pixel *P* che compongono ciascuna riga. Il numero di pixel e' **W**. Non sappiamo a priori quante righe vi sono nell'immagine, potenzialmente potremmo avere anche un numero infinito di righe. Quindi, altresì, non conosciamo il numero della colonne. Sappiamo però che lo stream di righe termina quando incontriamo il valore decimale -1.

Lo stream di dati è fornito tramite un file di testo che è posto in input al programma.

Il file è formattato secondo lo schema: `<valore_intero> \n`

Ad esempio il file sottostante indica:

Ogni riga è composta da 2 pixel RGBA, ossia W=2.

Scrivete il programma assembly MIPS che:

- legge il numero intero `0 <= W <= 100`
- legge la sequenza di pixel
- I valori letti in input sono tutti valori interi seguiti da accapo in modo che li possiate leggere con la syscall 5.

Esempio di input/output:

Input: (commentato, nei test NON ci saranno commenti)

```
2    # W
74   # R1  1st pixel - 1st row
235  # G1
232  # B1
89   # A1
201  # R2 2nd pixel
179  # G2
221  # B2
56   # A2
62   # R1 1st pixel - 2nd row (nota, non sappiamo quante righe abbiamo)
41   # G1
```

```
233  # B1
134  # A1
24   # R2 2nd pixel - 2nd row
104  # G2
171  # B2
117  # A2
-1   # end
```

Da notare che il file allegato `test-M00.in` è privo dei commenti:

Scrivete il programma assembly MIPS che: legge il numero intero 0 <= W < 100 legge la sequenza di pixel I valori letti in input sono tutti valori interi seguiti da accapo in modo che li possiate leggere con la syscall 5.

Il programma deve stampare una matrice lineare come quella in ingresso ma con un ordine diverso.

L'ordine della nuova matrice deve invertire la sequenza di ogni riga. Ossia stampare il seguente file, considerando l'esempio visto prima, deve generare la sequenza:

```
201  # R2 2nd pixel della matrice adesso appare in prima posizione.
179  # G2
221  # B2
56   # A2
74   # R1  1st pixel - 1st row, viceversa il primo pixel della prima matrice è ora in ultima p
235  # G1
232  # B1
89   # A1
24   # R2 2nd pixel - now 2nd row begins, 104  # G2
171  # B2
117  # A2
62   # R1 1st pixel - 2nd row (nota, non sappiamo quante righe abbiamo)
41   # G1
233  # B1
134  # A1
-1
```

Suggerimento: Potete stampare i numeri passo passo utilizzando le system calls:

- # print int, code $v0 ⟵ 1 , $a0 prende input
- # print char , code $v0 ⟵ 11, $a0 prende input per stampare il numero e il carattere newline.

Altro caso con 3 righe nel file `test-M-01.in`:

```
3
74    #1
235
232
89
201   #2
```

```
179
221
56
62     #3
41
233
134
24     #1
104
171
117
158    #2
104
91
72
96     #3
177
189
209
-1
```

Il programma deve generare:

```
62     # 3
41
233
134
201    #2
179
221
56
74     #1
235
232
89
96     #3
177
189
209
158    #2
104
91
72
24     #1
104
171
117
-1
```

Sono allegati i file di input del programma e anche il file di testo atteso (expected output). Dato che avete expected output potete anche debuggare il risultato usando diff sui sistemi Unix.

Per i sistemi Windows per la comparazione è possibile usare programmi grafici appositi oppure seguire questo link che usa Powershell https://serverfault.com/questions/5598/how-do-i-diff-two-text-files-in-windows-powershell

Per eseguire i test in locale:

- aprite una finestra di comando/console e posizionatevi nella directory in cui avete messo i file:
- il programma `program01.asm` da voi sviluppato
- il simulatore `Mars4_5.jar`
- un file di esempio di input `test-M-01.in`
- il corrispondente file di output atteso `test-M-01.out`

Eseguite in console il comando:

- `java -jar Mars4_5.jar me nc sm ic program01.asm < test-M-01.in > my_output-M-01.txt`
- in questo modo produrrete nel file `my_output-M-01.txt` le stampe del vostro programma
- confrontate il file `my_output-M-01.txt` con il file `test-M-01.out`, se sono uguali il test è superato

Trovate i 5 casi di test su cui allenarvi in allegato.

Per maggiori informazioni su come eseguire mars con input e output si veda:

https://q2a.di.uniroma1.it/21689/homework-test-homework?course=annunci/architetture-20-21

Come sempre scrivere nei commenti sotto se trovate bug o incongruenze che sistemiamo.