Homework 1 - sequenza a finestra scorrevole

Data una sequenza **S** di **N** numeri interi, da questa si può ottenere una sequenza derivata **Y** di **N-K** interi sommando, per ogni indice **i** nell'intervallo **[0 .. N-k-1]**, i **k** elementi di **S** agli indici **j** in **[i .. i+k-1]**.

Esempio (in cui evidenzio in **grassetto** un valore di Y ed i k elementi che sommati lo producono):

* k = 4
* S =   97, 77, 51, -96,   63, **45, -23, 26, -13**,   -42,    -30,   -95,    47,    -91, -12, -44, -10, 53, -31, -71
* Y = 129, 95, 63, -11, 111, **35**, -52, -59, -180, -120, -169, -151, -100, -157, -13, -32, -59

Non conoscendo a priori la lunghezza della sequenza S, non è possibile pre-allocare staticamente lo spazio di memoria necessario a memorizzarla completamente prima di calcolare la sequenza Y.

Se però si sa che **k** è limitato è possibile produrre la stampa della sequenza **Y** ricordando solo gli ultimi **k** elementi letti.

**Scrivete il programma assembly MIPS** che:

* legge il numero intero **0 < k < 21**
* legge la sequenza **S** di almeno **k** interi **diversi da 0** terminata dal valore 0 (zero, che non fa parte della sequenza)
  + mano a mano che ha informazioni sufficienti, stampa i valori della sequenza **Y** seguiti da accapo
* alla fine stampa (in quest'ordine, sempre seguiti da accapo) i valori
  + minimo della sequenza S
  + massimo della sequenza S
  + minimo della sequenza Y
  + massimo della sequenza Y

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)

*4                     # K   
97   
77   
51   
-96   
63   
45   
-23   
26   
-13   
-42   
-30   
-95   
47   
-91   
-12   
-44   
-10   
53   
-31   
-71   
0                 # indica fine della sequenza S, non fa parte della sequenza*

 Output atteso: (commentato, nei test NON ci saranno commenti)

*129   
95   
63   
-11   
111   
35   
-52   
-59   
-180   
-120   
-169   
-151   
-100   
-157   
-13   
-32   
-59   
-96             # minimo della sequenza S   
97              # massimo della sequenza S   
-180           # minimo della sequenza Y   
129            # massimo della sequenza Y*

**Consegna**

Il file da consegnare è il file sorgente **exercise01.asm** del vostro programma (meglio se commentato):

* **NON DEVE** contenere stampe/prompt diverse da quelle indicate
* **DEVE** iniziare la propria esecuzione dalla etichetta **main**
  + ricordatevi quindi di inserire la direttiva **.globl main**
* **DEVE** terminare la sua esecuzione con la **syscall 10**
* **NON DEVE** preallocare staticamente un vettore abbastanza grande da contenere tutta le sequenza S (per verificarlo verranno eseguiti dei test con grandi quantità di dati)
* NOTA: i compiti copiati vengono annullati

**Per consegnare** il file programma.asm sviluppato in Mars usate la [pagina di consegna](https://q2a.di.uniroma1.it/homeworks/delivery)

**Scadenza della consegna:**  **domenica 8 aprile ore 24**

**Per eseguire i test in locale**

* aprite una finestra di comando/console e posizionatevi nella directory in cui avete messo i file:
  + il programma **exercise01.asm** da voi sviluppato
  + il simulatore **Mars4\_5.jar**
  + un file di esempio di input **input.txt**
  + il corrispondente file di output atteso **expected.txt**

eseguite in console il comando

* java -jar Mars4\_5.jar me nc sm ic exercise01.asm < input.txt > output.txt
* in questo modo produrrete nel file output.txt le stampe del vostro programma
* confrontate il file output.txt con il file expected.txt, se sono uguali il test è superato

**File di test**

* [esempio.in](https://q2a.di.uniroma1.it/assets/architetture-17-18/HW1/esempio.in) (input) -> [esempio.out](https://q2a.di.uniroma1.it/assets/architetture-17-18/HW1/esempio.out) (expected)
* trovate altri test con i seguenti parametri nel file [tests-HW1.zip](https://q2a.di.uniroma1.it/assets/architetture-17-18/HW1/tests-HW1.zip)

|  |  |  |
| --- | --- | --- |
| **K** | **N** | **range di valori** |
| 1 | 1 | -1000 .. 1000 |
| 1 | 18 | -1000 .. 1000 |
| 2 | 26 | -1000 .. 1000 |
| 7 | 100 | -1000 .. 1000 |
| 8 | 57 | -1000 .. 1000 |
| 14 | 69 | -1000 .. 1000 |
| 17 | 62 | -1000 .. 1000 |
| 20 | 68 | -1000 .. 1000 |

**NOTA:** ho messo pochi dati piccoli per permettervi di debuggare il codice ma farò dei test anche con **N >> 1000** e con valori grandi

**Premi e cotillon**

Il voto di ciascun homework dà 0.5 o 1 punto (a seconda della difficoltà) da aggiungere ai voti delle prove scritte/ASM ed è composto da due parti:

* **correttezza** = % di test passati \* 0.5                 (da 0 a 0.5 per questo HW)
* **efficienza** = **+0.1** punti se siete nel primo 1/4 della classifica di efficienza di chi ha superato tutti i test

Per calcolare la classifica dell'efficienza uso il numero di istruzioni totali eseguite dal vostro programma sui test, in ordine crescente (lower = better).

PS. se qualcosa non è chiaro chiedete sotto