TestiGara - UndicesimaEdizione

Esercizi di Gara della XI edizione

AVVISI:

- Se non specificato altrimenti negli esercizi, le sequenze iniziali su nastro si intendono *non vuote*, ovvero contenenti almeno un simbolo.
- Per numero decimale si intende un numero positivo o nullo rappresentato con le cifre 0, 1, 2, ..., 9, senza zeri iniziali non significativi; per esempio 0 e 19 sono numeri validi, mentre 0032 deve essere scritto come 32.
- Nel fornire le soluzioni, ricordarsi di pulire il nastro finale da ogni simbolo che non costituisca la risposta!
- Ogni volta che si salva la soluzione di un esercizio con il simulatore della macchina di Turing, il "timestamp" dell'esercizio viene aggiornato con il tempo trascorso fino a quel momento. Si ricorda che a parità di punteggio, verrà considerato l'istante dell'ultimo salvataggio per ordinare la classifica finale.

Esercizio 1: Moltiplicatore per 10 [Punti 3]. Si scriva un programma per Macchina di Turing che, ricevuto sul nastro un numero decimale, lasci sul nastro alla fine della computazione il risultato della moltiplicazione del numero di ingresso per 10.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
3	30
450	4500
123456	1234560
0	0

Esercizio 2: Divisore per 10 (I) [Punti 5]. Si scriva un programma per Macchina di Turing che, ricevuto sul nastro un numero decimale, lasci sul nastro alla fine della computazione il risultato della divisione del numero di ingresso per 10, approssimato in basso.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
145	14
534623	53462
10	1
0	0

Esercizio 3: Riconoscimento numeri reali [Punti 10]. Si scriva un programma per Macchina di Turing che, ricevuta sul nastro una stringa qualunque sull'alfabeto { 0-9, +, -, .} lasci sul nastro la scritta SI se la stringa di ingresso rappresenta un numero reale in notazione decimale, definito come

[+ o -] <sequenza di cifre> [**.** <sequenza di cifre>]

Se la stringa di ingresso non è un numero reale in notazione decimale, il programma deve lasciare sul nastro la stringa NO. Si noti che, sotto l'assunzione che l'input sia finito, tutti i numeri esprimibili in questa notazione sono in realtà numeri razionali: come tutti i sistemi di elaborazione digitali, la Macchina di Turing può solo trattare approssimazioni razionali dei numeri reali.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
34	SI
-342.123	SI
-343-342	NO
234.234.23	NO
+12.45	SI
-38	SI
--34	NO
0	SI

Esercizio 4: Divisore per 10 (II) [Punti 12]. Si scriva un programma analogo a quello dell'esercizio 2, in grado però di operare su numeri reali (con opzionalmente segno e parti decimali) e approssimante verso lo 0. Il segno va preservato nel risultato.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
145	14
-3017	-301
-3.1415926	0
31.415926	3
-100.5	-10

Esercizio 5: Divisore per 10 (III) [Punti 15]. Si scriva un programma analogo a quello dell'esercizio 4, che approssimi il risultato all'intero più vicino anziché verso lo 0.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
145	14
145.1	15
137.81	14
-218.0001	-22
5	0
6	1

Esercizio 6: Conteggio di sottostringhe [Punti 25]. Si scriva il programma di una Macchina di Turing che, date sul nastro due stringhe formate da lettere nell'alfabeto { A..J } separate da $, conta le occorrenze distinte della stringa a sinistra del $ all'interno di quella a destra del $. Il programma deve lasciare sul nastro il numero di occorrenze trovate, espresso in notazione decimale.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
AB$ABBACCHIA	1
BA$BABA	2
AB$CABABBAABDABBBABABBBBE	6
AA$AAABBAA	2
F$ABCDE	0
AAA$AAABBAA	1

Esercizio 7: Percentuale [Punti 25]. Si scriva un programma per Macchina di Turing che, ricevuta in ingresso una stringa della forma *a***%***b*, in cui *a* e *b* sono numeri decimali e *a* è compreso fra 0 e 100 (inclusi), lasci sul nastro al termine della computazione il numero naturale corrispondente all'*a*% di *b*, approssimato in basso.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
10%50	5
25%100	25
33%1000	333
4%1890	75
0%3240	0
100%12	12
50%0	0

Esercizio 8: Parole simili [Punti 30]. Definiamo la *distanza* tra due parole composte di lettere A..Z come il numero di posizioni in cui le lettere corrispondenti nelle due parole differiscono. Nel caso una parola sia più corta dell'altra, si assume che essa differisca in tutte le posizioni rimaste. Per esempio, ABACO e ABATE sono a distanza 2, mentre ACANTO e ABATE sono a distanza 4. Una nozione di distanza simile a questa (ma leggermente più complessa) è usata, fra l'altro, dai correttori ortografici forniti con i programmi di videoscrittura.

Si scriva il programma di una macchina di Turing che, dato sul nastro di input una parola seguita dal simbolo **$** e una lista di parole separate da **:**, lasci sul nastro alla fine della computazione la parola della lista più "vicina" alla prima (ovvero, la cui distanza rispetto alla prima parola è minima). Nel caso vi siano più parole con pari distanza minima, il programma dovrà restituire come risultato la prima della lista (ovvero, quella più a sinistra). Si assuma che ciascuna parola sia lunga al più 9 caratteri.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
PEFA$PELO:PERA:PASSA	PERA
PASSARE$FIUTARE:PASSATA:PASSARE	PASSARE
NULLA$	
SQUOLA$SQUALO:SCUOLA:STUOLO:STOLA	SCUOLA

Esercizio 9: La prova del 9 [Punti 42]. La prova del 9 è un metodo per verificare la correttezza di una moltiplicazione, basato sulle proprietà dell'aritmetica modulare. Il metodo di prova si basa sul fatto che per ogni n, $10^n \bmod 9 = 1$; tecnicamente, diciamo che la somma delle cifre di un numero decimale mantiene l'equivalenza al numero in Z_9 (l'anello degli interi modulo 9).

La forma tradizionale della prova del 9 è la seguente. Innanzitutto si traccia una croce, che delimita

quattro spazi che chiameremo A, B, C e D:

A	B
C	D

Nelle quattro zone si scrive (volendo verificare per esempio se è corretto il risultato 1902×1964=3735528):

1. In A: si sommano ripetutamente le cifre del moltiplicando, il primo fattore, finché non resta un numero ad una sola cifra: es. 1902 => 1+9+0+2=12; 12 => 1+2=3
2. In B: si applica lo stesso procedimento con il moltiplicatore, il secondo fattore: es. 1964 => 1+9+6+4=20; 20 => 2+0=2
3. In C: si moltiplicano i 2 numeri appena ottenuti e posti in alto sulla croce. Se il risultato della moltiplicazione ha più cifre, esse sono ripetutamente sommate come prima: es. 3*2=6.
4. In D: si sommano le cifre del risultato presunto della moltiplicazione: es. 3735528 => 3+7+3+5+5+2+8=33; 33 => 3+3=6

Nel caso del nostro esempio avremo dunque:

3	2
6	6

Quando, come in questo caso, i due numeri in basso sono uguali allora la prova ha esito positivo, altrimenti ha esito negativo.

Si noti però che:

- se la prova ha esito negativo allora la moltiplicazione è **sicuramente** errata
- se la prova ha esito positivo, il risultato trovato **potrebbe** essere errato, e differire dal risultato reale per un multiplo di 9. Infatti, se al risultato si sommasse o sottraesse 9 o un suo multiplo, il test sarebbe ancora superato (9 è infatti equivalente a 0 in Z_9)!

Si scriva un programma per Macchina di Turing che, ricevuta sul nastro una moltiplicazione con un risultato presunto, nella forma *a***b*=*c* (con *a*, *b* e *c* numeri decimali) lasci sul nastro la stringa SI se la prova del 9 ha esito positivo, NO in caso contrario. Come già osservato, il programma deve rispondere SI anche se la moltiplicazione è errata, ma la prova del 9 è superata.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
123*456=56088	SI
123*456=56089	NO
123*456=56097	SI
0*12=0	SI
150*30=4500	SI
150*30=9	SI
150*30=4300	NO

Esercizio 10: Espressioni regolari [Punti 45]. Una *espressione regolare* identifica un insieme di stringhe, ovvero un *linguaggio*. Un'espressione regolare è denotata da un *pattern*, ovvero da una sequenza

di caratteri, alcuni dei quali dotati di una particolare interpretazione, che indica quali stringhe appartengono all'insieme e quali no. Ad esempio il pattern CIAO+ denota l'insieme di tutte le stringhe formate dai caratteri 'C', 'I', 'A' e che terminano con una sequenza di uno o più caratteri 'O'. Ecco quindi che le stringhe CIAO, CIAOO, CIAOOOOO ecc. apparterranno all'insieme (si usa dire che esse *soddisfano* l'espressione regolare), mentre XX, CIA, CIAOA ecc. non vi apparterranno.

Consideriamo pattern formati da sequenze di caratteri da interpretare come segue:

- Ogni carattere nell'alfabeto { A..Z } indica che nella stringa da verificare deve essere presente lo stesso carattere
- Il carattere '.' indica che nella stringa da verificare deve essere presente un carattere qualsiasi
- Se un carattere è seguito dal carattere '*' si intende che nella stringa da verificare il carattere precedente l'* può occorrere 0 o più volte (es. CI*AO indica le stringhe della forma CAO, CIAO, CIIAO, ...)
- Se un carattere è seguito dal carattere '+' si intende che nella stringa da verificare il carattere precedente il + può occorrere 1 o più volte (es. CI+AO indica le stringhe della forma CIAO, CIIAO, ...). Si noti che per ogni carattere *c*, il pattern *c*+ è equivalente al pattern *cc**.

Si scriva il programma di una Macchina di Turing che data sul nastro un pattern nel formato descritto, seguito dal simbolo $ di separazione e da una stringa (anche vuota) sull'alfabeto { A..Z }, lasci sul nastro SI se la stringa soddisfa l'espressione regolare denotata dal pattern, NO altrimenti.

Esempi:

NASTRO INIZIALE	*NASTRO FINALE*
CIAO$CIAOO	NO
CIAO$CIAO	SI
C*I*A*O*$	SI
C*I*A*O*$IIAAA	SI
C*I*A*O*$ICIAO	NO
C+I*AO$CIAO	SI
C+I*AO$CCIIIAO	SI
C+I*AO$CIAOO	NO
C.AO$CIAO	SI
C.AO$CEAO	SI
C.AO$EEAO	NO
.*$	SI
.*$BUONLAVORO	SI
.*A$BUONLAVORO	NO