Programmazione I
Prova di programmazione – 15 Giugno 2023 – 2 ore

Partendo dal frammento di codice fornito, realizzare un controllore simulato di una coppia di nastri trasportatori collocati uno affianco all'altro (in parallelo), e tali che gli oggetti trasportati si possano spostare da un nastro all'altro. Ci riferiremo a ciascuno dei nastri come il *nastro 0* o *in alto* ed il *nastro 1* o *in basso*. I due nastri sono composti da una sequenza lineare di *sezioni* adiacenti. Ogni sezione può scorrere in una delle seguenti quattro direzioni ed è rappresentata dal simbolo riportato di fianco: destra **>**, sinistra **<**, alto **^**, basso **v**. Il numero di sezioni è uguale per entrambi i nastri. Nella descrizione della funzionalità 2 riportiamo un esempio. All'avvio del sistema i due nastri hanno 0 sezioni. Il programma fornisce le seguenti funzionalità.

1. **[+2] inizializza_nastri(s1, s2)** Si inizializzano i nastri per contenere le sezioni rappresentate, mediante i simboli definiti sopra, e senza spazi, nelle stringhe **s1** ed **s2**. Le due sezioni devono avere la stessa lunghezza.. Si ottiene il punteggio aggiuntivo massimo se si evitano deallocazioni e riallocazioni di memoria non necessarie. Per il collaudo, fare leggere dal programma prima la lunghezza *M* delle stringhe e poi le stringhe stesse. Se *M*>0 allora le stringhe vanno lette per intero anche se contengono caratteri scorretti o hanno lunghezza inferiore ad M. L'aggiornamento dei nastri invece va effettuato solo se le stringhe sono corrette.
2. **stampa_nastri** Stampa la composizione dei due nastri su *stdout*. Ad esempio, stampa

```
>>v<>
<^>v^
```

nel caso di due nastri con 5 sezioni, in cui, se si colloca un oggetto, ad esempio, all'inizio del nastro in alto, tale oggetto sarà trasportato a destra per due sezioni, poi sarà spostato sul nastro in basso, il quale lo trasporterà a destra di una sezione e poi lo farà uscire verso il basso. Altro esempio: se si colloca un oggetto sull'ultima sezione del nastro in alto, tale oggetto esce direttamente dal nastro verso destra.
3. **[2, +1] salva_nastri** Salva la composizione dei nastri in un file di testo. Punteggio aggiuntivo se si evita replicazione del codice.
4. **[2, +2] carica_controllo** Carica la composizione dei nastri dal file. L'eventuale precedente composizione è persa. Si ottiene il punteggio aggiuntivo massimo se si evitano deallocazioni e riallocazioni di memoria non necessarie e si evita replicazione del codice.
5. **[2] ritorna_nuova_posizione(n, s)** Simula il trasporto di un oggetto da parte della sezione di indice **s**, sul nastro **n** (0 in alto, 1 in basso). Assumiamo che le sezioni siano numerate con indici crescenti verso destra, a partire da 0. In particolare, se gli indici sono corretti e la sezione trasporta l'oggetto su un'altra sezione di uno dei due nastri, allora ritorna gli indici del nuovo nastro e della nuova sezione. Altrimenti ritorna falso. Per collaudare questa funzionalità, fare stampare gli indici nel primo caso, e la stringa **`FUORI`** nel secondo caso. Ad esempio, per i nastri mostrati nella funzionalità 2, si deve stampare: **`0 1`** se si passa (0, 0), **`1 2`** se si passa (0, 2), oppure **`FUORI`** se si passa (1, 0) o (0, 4) o (1, 3).
6. **[4] trasporta_oggetto(n, s)** Simula il trasporto completo di un oggetto da parte di tutte le sezioni una dopo l'altra, nelle stesse ipotesi della funzionalità precedente. In particolare, stampa l'indice dell'ultimo nastro e dell'ultima sezione su cui passa l'oggetto prima di essere fatto uscire. Oppure stampa **`LOOP`** se l'oggetto non riesce più ad uscire dai nastri. Infine stampa **`FUORI`** se gli indici passati non sono correttiAd esempio, dati i seguenti nastri:

```
>v<<>^
^>>^v<
```

la funzionalità stampa le seguenti stringhe: **`0 5`** se le si passa (0, 4) oppure (0, 5), **`1 4`** se le si passa (1, 4) oppure (1, 5), **LOOP** in tutti gli altri casi.

I parametri di ingresso delle funzionalità sono solo indicativi. Gestire opportunamente le situazioni di errore, tranne l'*overflow* e l'inserimento di dati in formato errato da *stdin*.

Per il collaudo: se fate stampare messaggi per invitare l'utente ad inserire dell'input, ricordate di aggiungere la stampa di caratteri accapo. Altrimenti nel puro output del programma vi saranno delle righe fuse, e di fatto tale output non sarà quello che credete (le righe fuse non le vedete quando usate il programma da terminale, perché inserite voi l'accapo da utenti).

REGOLE

- Si può utilizzare ogni genere di manuale e di materiale didattico
- Per superare la prova, bisogna svolgere almeno i punti 1 e 2. Se si svolgono solo tali punti, il programma deve essere perfettamente funzionante. Il voto ottenuto in questo caso è 18.
- Ciascuna funzionalità DEVE essere implementata mediante almeno una funzione.
- Il voto massimo si ottiene se
  a) il programma è perfettamente funzionante in ogni sua parte
  b) tutti i principi di ingegneria del codice visti nel corso sono stati applicati
  c) sono state seguite eventuali altre indicazioni presenti nella traccia in merito al voto finale