Programmazione I
Prova di programmazione – 26 Gennaio 2023 – 2 ore e mezza

Partendo dal frammento di codice fornito, realizzare un programma per gestire sequenze di spostamenti di un robot su un pavimento. Il pavimento è modellato come una griglia di posizioni, che partono da (0, 0), hanno solo coordinate positive e vanno verso il basso (in avanti) e verso destra per valori crescenti della prima e della seconda componente. Ogni spostamento nella sequenza è di una posizione, in avanti, indietro, a destra o a sinistra. Ad esempio, dalla posizione (0,0), uno spostamento in avanti porta nella posizione (1, 0). All'avvio del programma, la sequenza è vuota. Il programma fornisce le seguenti funzionalità.

1. **[+2] inizializza_sequenza(N)** Reinizializza la sequenza a contenere **N** spostamenti. Se **N** > 0, gli spostamenti sono letti da *stdin*, sotto forma di caratteri: **<** a sinistra, **>** a destra, **v** in avanti, ^ indietro. Un esempio è riportato nella descrizione della seguente funzionalità. Se il formato della sequenza inserita da *stdin* non è valido, allora la sequenza originale memorizzata dal programma rimane invariata (si legge comunque tutta la sequenza di caratteri su *stdin*, per svuotare l'input). Per semplicità, assumere che la sequenza non porterà mai il robot in una posizione con qualche coordinata negativa. Si ottiene il punteggio aggiuntivo se si evitano deallocazioni e riallocazioni di memoria non necessarie.
2. **stampa_sequenza** Stampa la sequenza di spostamenti, rappresentando ciascun passo come nel precedente punto. Ecco un esempio di stampa, in cui il robot va avanti, indietro e poi due volte a destra:

```
v       ^       >       >
```

3. **[2, +1] salva_sequenza** Salva la sequenza in un file di testo dal nome predefinito. Punteggio aggiuntivo se si evita replicazione del codice.
4. **[2, +2] carica_sequenza** Carica la sequenza dal file. L'eventuale precedente sequenza 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. **[3] stampa_sequenza2** Stampa la sequenza di spostamenti come nel punto 2, ma aggiunge una riga contenente le nuove posizioni occupate dal robot a seguito di ciascuno spostamento. Si assume che il robot parta dalla posizione (0,0). In particolare, per ciascuna posizione stampa le coordinate tra parentesi tonde. Ecco un esempio di stampa:

```
v       ^       >       >
(1 0)   (0 0)   (0 1)   (0 2)
```

6. **[3] stampa_traccia()** Stampa una traccia delle posizioni occupate dal robot, rappresentando ciascuna posizione occupata con un asterisco e ciascuna posizione non occupata con un -, ed assumendo che il robot parta da (0, 0). Ad esempio, per le posizioni mostrate nel precedente esempio, stampa:

```
* * *
* - -
```

Può essere utile considerare che si può implementare una matrice dinamica di *KxM* elementi mediante un array dinamico lineare di *KxM* elementi, e che dati gli indici *(i, j)* di un elemento della matrice, si ottiene l'indice dello stesso elemento, nell'array sottostante, con la formula *i*M+j*.

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