Programmazione I
Prova di programmazione – 12 Giugno 2019 – 2 ore

Partendo dal frammento di codice fornito, realizzare un programma per la gestione di alcune proprietà di un insieme di processi in esecuzione in un sistema operativo. Ogni processo è identificato da un numero intero positivo (*pid*), e può essere in stato *funzionante* o *fallito*. Infine un processo, diciamo *p1*, ha una *sincronizzazione* rispetto a qualche altro processo nell'insieme, diciamo *p2*, se *p1* ogni tanto si blocca e, per riprendere la sua esecuzione, deve aspettare che *p2* esegua determinate istruzioni (non specificate in questa traccia). Non è ammessa una sincronizzazione rispetto a processi in stato fallito. Infine un processo può essere sincronizzato al massimo rispetto ad un solo altro processo. All'avvio del programma l'insieme dei processi è vuoto. Il programma fornisce le seguenti funzionalità.

1. **inizializza_insieme(N)** Inizializza l'insieme dei processi a contenere *N* processi, con *pid* progressivi a partire da 0. L'eventuale contenuto precedente dell'insieme è perso. I processi sono tutti in stato funzionante, e non hanno sincronizzazioni.
2. **stampa_insieme** Stampa l'insieme dei processi. Un esempio di output per un insieme di 4 processi appena inizializzato potrebbe essere
   ```
   0  funzionante     no-sinc
   1  funzionante     no-sinc
   2  funzionante     no-sinc
   3  funzionante     no-sinc
   ```
3. **salva_insieme** Salva l'insieme dei processi in un file di testo dal nome predefinito.
4. **carica_insieme** Carica l'insieme dal file. L'eventuale precedente contenuto dell'insieme è perso.
5. **crea_sincronizzazione(p1, p2**) Imposta il processo **p1** come sincronizzato rispetto al processo **p2**, a patto che il processo **p1** non sia già sincronizzato con alcun altro processo e che il processo **p2** non sia in stato fallito. Ad esempio, se si crea una sincronizzazione per i processi 1 e 2 dell'esempio al punto 2, e si sincronizzano entrambi i processi rispetto al processo 3, allora lo stato dell'insieme dei processi diviene:
   ```
   0  funzionante     no-sinc
   1  funzionante     3
   2  funzionante     3
   3  funzionante     no-sinc
   ```
6. **imposta_fallito(p)** Imposta il processo **p** in stato fallito, eliminando eventuali sincronizzazioni di altri processi rispetto a **p**. Ad esempio, se si impostano a fallito i processi 1e 3 per l'insieme di cui all'esempio al punto 5, allora lo stato dell'insieme diviene:
   ```
   0  funzionante     no-sinc
   1  fallito         no-sinc
   2  funzionante     no-sinc
   3  fallito         no-sinc
   ```
   Si ottiene il punteggio massimo se si realizza questa funzionalità senza visitare tutto l'insieme dei processi quando si aggiornano le sincronizzazioni. Non preoccuparsi dell'eventuale spazio aggiuntivo occupato per realizzare una tale soluzione efficiente. Notare che se la soluzione aggiunge stato, allora tale stato aggiuntivo potrebbe necessitare di un corretto salvataggio e caricamento. Si ottiene un punteggio più alto anche se non si riesce a gestire anche questo aspetto.

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*.

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 (almeno 30) 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