Programmazione I
Prova di programmazione – 14 giugno 2018 – 2 ore

Partendo dal frammento di codice fornito, realizzate un programma di gestione delle dipendenze per un insieme di pacchetti software. Diciamo che un pacchetto **p1** dipende da un altro pacchetto **p2**, se il pacchetto **p2** deve essere presente nel sistema affinché il pacchetto **p1** possa essere utilizzato. Assumiamo che ciascun pacchetto sia identificato da una lettera maiuscola progressiva, a partire dalla *A*, e, in quanto alle dipendenze, che il pacchetto possa essere in uno di due possibili stati: non dipendente da alcun pacchetto, oppure dipendente da uno degli altri pacchetti nell'insieme. Chiamiamo *insieme delle dipendenze* l'insieme degli stati dei pacchetti. La dimensione massima per l'insieme di pacchetti, e quindi delle dipendenze, è 20. Nel programma, l'insieme deve occupare **solo la memoria necessaria** per memorizzare gli stati che contiene. All'avvio del programma, l'insieme è vuoto. Il programma deve fornire le seguenti funzionalità.

1. **reinizializza_insieme(N)** Inizializza l'insieme delle dipendenze a contenere lo stato di **N** pacchetti, dal pacchetto *A* a quello identificato dall'*N*-esima lettera dell'alfabeto. Lo stato di ciascun pacchetto è "non dipendente da alcun pacchetto". L'eventuale insieme precedente è perso.
2. **imposta_dipendenza(p1, p2)** Imposta lo stato del pacchetto **p1** come dipendente dal pacchetto **p2**. Entrambi i pacchetti sono individuati attraverso la lettera che li identifica. Il precedente stato del pacchetto **p1** è perso. Ad esempio, se si passano i pacchetti *D* ed *E* alla funzionalità, allora lo stato del pacchetto *D* viene impostato come dipendente dal pacchetto *E*.
3. **stampa_dipendenze** Stampa tutti gli stati nell'insieme delle dipendenze. In particolare, stampa solo l'identificatore del pacchetto per ciascun pacchetto senza dipendenze, mentre, per ciascun pacchetto **p** con una dipendenza, stampa una freccia e l'identificatore del pacchetto da cui **p** dipende. Ecco un esempio di stampa, per un insieme di stati di sei pacchetti:
   ```
   A -> F
   B -> C
   C
   D -> A
   E -> A
   F -> E
   ```
   Notate che in questo insieme di dipendenze c'è un ciclo: $A \rightarrow F \rightarrow E \rightarrow A$.
4. **salva_insieme** Salva l'insieme delle dipendenze, in un file **binario** dal nome definito a tempo di scrittura del programma. NOTA: una **`write`** di una struct contenente un puntatore ad un array, salva il contenuto del puntatore e non dell'array.
5. **carica_insieme** Carica l'insieme delle dipendenze. L'eventuale precedente contenuto è perso.
6. **stampa_catena_dipendenze(p, n)** Stampa la catena di al massimo **n** dipendenze che parte dal pacchetto **p**. Alcuni esempi, in base ai parametri passati in ingresso, e per l'insieme d'esempio per il punto 3:
   (C, 4): **`C`** (B, 3): **`B -> C`** (D, 3): **`D -> A -> F -> E`**

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