Programmazione I
Prova di programmazione – 2 Settembre 2021 – 2 ore

Partendo dal frammento di codice fornito, realizzare un allocatore di memoria. La memoria è intesa come sequenza di celle contigue. All'interno di tale memoria, l'allocatore alloca degli *oggetti*, ossia riserva, per tali oggetti, delle porzioni contigue di memoria. Se una cella appartiene ad una porzione di memoria riservata ad un oggetto, allora tale cella si considera occupata. Altrimenti la cella è libera. All'avvio dell'allocatore, la memoria è vuota, ossia contiene zero celle. L'allocatore fornisce le seguenti funzionalità.

1. **inizializza_memoria(N)** Inizializza la memoria a dimensione **N**. La memoria è tutta libera. L'eventuale precedente contenuto della memoria è perso.
2. **stampa_memoria** Stampa il contenuto della memoria col seguente formato. Data la dimensione N della memoria, stampa una riga di N cifre in base 10, in cui la cifra nella posizione *i*-esima ha valore *i*%10. Sotto tale riga, stampa una seconda riga, in cui il carattere *i*-esimo è, rispettivamente, un trattino o un asterisco se la cella nella posizione *i*-esima è libera o occupata. Ad esempio, se N=16 e ci sono tre oggetti nelle posizioni 2, 7 e 14 (e tali oggetti hanno dimensioni 3, 4 ed 1 rispettivamente), allora stampa:
   ```
   0123456789012345
   --***--****---*-
   ```
   Ovviamente per poter avere oggetti presenti in memoria, bisogna aver prima implementato la funzionalità seguente.
3. **[3, +2] alloca_oggetto(pos, dim)** Alloca un oggetto di dimensioni **dim** nella posizione **pos** all'interno della memoria. Le posizioni partono da 0. Se vi sono **dim** celle libere a partire dalla posizione **pos**, l'operazione ha successo, e tali celle passeranno dallo stato libero allo stato occupato. Altrimenti l'operazione fallisce e nessuna nuova cella passa allo stato occupato. Si ottiene il punteggio massimo se si realizza questa funzionalità a costo lineare rispetto al numero di oggetti allocati, e non rispetto al numero di celle di memoria. Ad esempio, partendo dallo stato riportato nell'esempio, al punto 2, se si alloca un oggetto in posizione 12 e di dimensione 1, il nuovo stato diventa:
   ```
   0123456789012345
   --***--****-*-*-
   ```
   L'operazione invece fallisce, e lo stato non cambia, se la dimensione è ad esempio maggiore di 2.
4. **[2] salva_memoria** Salva il contenuto della memoria in un file di testo.
5. **[3] carica_memoria** Carica il contenuto della memoria dal file di testo. Il precedente contenuto è perso.
6. **[3, +2] alloca_oggetto2(pos, dim)** Svolge le stesse operazioni della funzionalità 2, ma, in aggiunta, sposta in avanti l'eventuale oggetto presente nelle posizioni uguali o successive a **pos**, nel caso in cui questo sia necessario e sufficiente per riuscire ad allocare anche il nuovo oggetto. Si ottiene il punteggio massimo se si realizza questa funzionalità a costo lineare rispetto al numero di oggetti allocati, e non rispetto al numero di celle di memoria.

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 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