Prova Pratica Programmazione 1, 23/12/2021

Partendo dal frammento di codice fornito, realizzare un programma per gestire i voli in transito e gli aerei all'interno di un aeroporto. Ogni aereo dovrà essere identificabile tramite un ID di volo alfanumerico di al massimo 10 caratteri. Per ogni aereo in transito, l'aeroporto dovrà gestire le piste e gli aerei che transitano in tali piste. L'aeroporto ha M piste indicizzate da 0 a M-1, all'interno di ogni pista vi possono transitare al più k aerei con k definito come 3.

Il programma fornisce le seguenti funzionalità:

1. **Inizializza_Aeroporto(M)** Inizializza tutte le strutture dati necessarie e le piste che l'aeroporto gestirà. L'eventuale contenuto precedente dell'aeroporto viene perso.
2. **[+2] Aggiungi_aereo_a_pista(ID_ Aereo, NR_pista)** tale funzionalità proverà ad aggiungere alla pista **NR_pista** l'aereo con **ID_Aereo**. Nel caso in cui non vi sia spazio per questo aereo sulla pista selezionata, tale aereo viene dirottato su una delle prime piste in cui vi sia spazio, ovvero, poiché le piste sono indicizzate da 0 a M-1, trovare uno spazio disponibile per l'aereo partendo dalla pista con indice 0. Nel caso limite in cui tutte le piste siano già piene, la funzionalità ritorna falso (vero in caso di successo). Se quest'ultimo controllo è implementato in O(1) (tempo costante) rispetto al numero di aerei si ottiene il punteggio aggiuntivo.
3. **Stampa_stato_piste()** tale funzionalità permette di visualizzare, per ogni pista, la situazione corrente dei voli. Stampare solo numero della pista e gli ID alfanumerici dell'aereo in transito su quella pista come illustra l'esempio di output.
   Esempio:
   0 - A123 B122 C212
   1 - S11 K999 .
   2 - . . M3212
   3 - . . .
   …
   Usare il carattere di spaziatura per separare i voli e seguire il template fornito come da esempio.

   [NR_PISTA – ID_1 ID_2 ID_3]
   Nel caso in cui l'aereo non sia presente in una determinata posizione, stampare un "**.**"
   Ricordarsi inoltre che il numero della prima pista è 0.

4. **[2] Salva_stato()** salva lo stato corrente dell'aeroporto in un file di testo dal nome predefinito.
5. **[3] Carica_stato()** carica uno stato precedentemente salvato dell'aeroporto. Il procedente stato dell'aeroporto è perso.
6. **[3 +1] Rimuovi_aereo(ID_Aereo)** Tale funzionalità dovrà ricercare in che pista risiede l'aereo ed andare a rimuoverlo facendo spazio per un ipotetico nuovo aereo in transito nell'aeroporto. La funzionalità ritorna true se l'aereo è stato trovato e rimosso, false in caso contrario. Il punteggio aggiuntivo si ottiene tramite il completamento dell'implementazione richiesta al punto 2.

7. **[4] Compatta_piste()** Tale funzionalità fa avanzare tutti gli aerei che sono in ogni pista nelle posizioni libere di indice minore della stessa pista. Applicando questa funzione all'output presentato nel punto 3 si otterrebbe:
   0 - A123 B122 C212
   1 - S11 K999 .
   2 - M3212 . .
   3 - . . .

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, 2 e 3. 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