Compito di Programmazione
27 Maggio 2022

PER GLI STUDENTI ONLINE:
alla fine inviare email con pdf dell'esame a
cosimo.laneve@unibo.it
giuseppe.lisanti@unibo.it
adele.veschetti2@unibo.it

Nota Bene. Ogni esercizio deve essere svolto su una pagina diversa.

1. **(punti 8)** Un nodo è un elemento il cui tipo è `struct nodo { int val; int next ; } ;`.

   Una `lista_di_nodi` è implementata attraverso un array di elementi `nodo` la cui testa si trova ad indice 0 e il cui campo `next` indica la posizione dell'elemento successivo. Se un nodo ha `next == -1` allora quel nodo è finale (non c'è alcun elemento successivo). Ad esempio

1; 4		3; -1		5; 2
0	1	2	3	4

   implementa la lista `1.5.3`. Si osservi che i nodi non sono memorizzati in maniera contigua.

   Definire una funzione `revert` che prende una `lista_di_nodi A` (quindi implementata come array) **non vuota** e modifica `A` in maniera tale che rappresenti la lista invertita. [Attenzione al vincolo che l'elemento ad indice 0 di `A` deve rappresentare la testa della lista; punti max 8 se la funzione è ricorsiva; punti max 6 se la funzione è iterativa.]

2. **(punti 8)** Un'azienda ferroviaria con treni ad alta velocità utilizza una lista per memorizzare i viaggi che offre. Ogni viaggio deve memorizzare la stazione di partenza, quella di arrivo, il prezzo del biglietto, il numero dei posti disponibili e il numero di cambi necessari. Non è possibile che ci siano più viaggi tra le stesse stazioni. Definire le strutture dati della lista dei viaggi e le seguenti funzioni:

   - `cerca_viaggio` che prende come parametri la stazione di partenza, quella di arrivo, il numero di posti che si vuole prenotare e il numero massimo di cambi che si è disposti a fare. La funzione ritorna il prezzo complessivo da pagare, se esiste un viaggio che soddisfa le richieste dell'utente.
   - `aggiungi_viaggio` che prende come parametri tutte le informazioni del nuovo viaggio. La funzione aggiunge **in coda** alla lista il nuovo viaggio, se non è già presente nella lista.

3. **(punti 8)** Una `Conferenza` è caratterizzata da un'orario di inizio, un'orario di fine e un'orario in cui si svolge la pausa pranzo. Inoltre è caratterizzata da un insieme di incontri, ognuno dei quali ha un orario in cui si svolge, il titolo dell'incontro e il nome del relatore (ogni incontro dura un'ora). Si rappresentino il singolo incontro e l'insieme di incontri (non è possibile utilizzare le liste) e si implementino il costruttore e i seguenti metodi:

   - `num_slot_disponibili`: restituisce il numero di slot di tempo non ancora utilizzati.
   - `controlla_orario`: riceve l'orario per l'incontro e verifica che rispetti gli orari della conferenza.
   - `controlla_slot`: riceve l'orario per l'incontro e verifica che non sia preso da un altro incontro.
   - `aggiungi_incontro`: riceve le informazioni di un incontro e si avvale dei metodi sopra per verificare se è possibile o meno aggiungere l'incontro, in caso positivo lo inserisce.

   Successivamente si implementi la classe `ConferenzaPremium` la quale prevede anche uno slot orario per la presentazione di poster. Durante la presentazione dei poster non si possono svolgere incontri.
   Si implementi il costruttore e si sfrutti l'ereditarietà per modificare solo i metodi che è necessario modificare per tenere conto dell'orario dei poster.