(Esercizi Tratti da Temi d'esame degli ordinamenti precedenti)

Esercizio 1

L'agenzia viaggi GV - Grandi Viaggi vi commissiona l'implementazione della funzione `AssegnaVolo`. Tale funzione riceve due liste dinamiche, una riguardante i voli a disposizione, con i campi data, ora, numero di volo, aeroporto origine, aeroporto destinazione, posti disponibili per l'agenzia. La seconda lista descrive i clienti che hanno acquistato un pacchetto vacanze, con i campi nome, cognome, data partenza, luogo della vacanza, numero partecipanti, puntatore al volo; quest'ultimo vale NULL se al cliente non è stato ancora assegnato un volo.

La funzione `AssegnaVolo` deve associare per ciascun cliente l'opportuno volo, tenendo conto della data di partenza e della destinazione della vacanza. Per ciascun volo della lista voli con un numero di posti disponibili maggiore di zero, si assegneranno i viaggi non ancora assegnati, così che l'assegnazione a uno specifico volo terminerà quando i posti saranno tutti esauriti. L'assegnazione di un volo al cliente consiste nell'assegnare allo specifico campo nella lista dei clienti l'indirizzo del nodo che descrive il volo assegnato, decrementando il numero di posti disponibili nel nodo che descrive il volo. Il processo termina quando non ci sono più clienti senza volo assegnato o voli con posti disponibili.

Dopo aver dichiarato un'ipotetica struttura dati con almeno i campi essenziali di ciascuna lista, implementare la funzione richiesta in C/C++.

Esercizio 2

Il magazzino della EmmePi SpA è suddiviso in settori, in base al reparto in cui i beni sono impiegati. Vi viene commissionata l'implementazione della funzione `GiacenzaElevata` che riceve come parametro il nome del reparto e due liste. La prima lista contiene il codice del prodotto (stringa di 15 caratteri), il nome del reparto di appartenenza e il prezzo del prodotto. La seconda lista contiene il codice del prodotto (stringa di 15 caratteri), il codice del magazzino (numerico) e la giacenza, nel magazzino (un prodotto può essere presente in più magazzini).

La funzione restituisce il numero di prodotti del reparto indicato che hanno una giacenza complessiva superiore alle 100 unità e una giacenza minima in ciascun magazzino superiore alle 10 unità.
Definire le strutture dati delle due liste e implementare la funzione in C/C++.

Esercizio 3

Il magazzino della EmmePi SpA è suddiviso in settori, in base al reparto in cui i beni sono impiegati. Vi viene commissionata l'implementazione della funzione `ValoreReparto` che riceve come parametro il nome del reparto e due liste. La prima lista contiene il codice del prodotto (stringa di 15 caratteri), il nome del reparto di appartenenza e il prezzo del prodotto. La seconda lista contiene il codice del prodotto (stringa di 15 caratteri) e la giacenza, ossia la quantità residua in magazzino.

La funzione elenca a schermo tutti i prodotti che fanno riferimento al reparto passato come parametro e restituisce al chiamante il valore (prezzo per quantità) di tali prodotti. Definire le strutture dati delle due liste e implementare la funzione in C/C++.

Esercizio 4

Si definisca la struttura dati che definisce una lista dinamica di nominativi (lunghezza massima di 100 caratteri), dove ogni nominativo ha associato un codice numerico (intero).

Si consideri anche un file che contiene un elenco di nominativi, strutturato nel modo seguente: ogni riga del file contiene i dati di un nominativo, dove i primi 6 caratteri contengono il codice del nominativo, mentre i restanti 100 contengono il nominativo vero e proprio.

Si scriva quindi la funzione `CreazioneListaDaFile` che restituisce l'indirizzo della testa di una lista di nominativi (definita come detto precedentemente). La funzione riceve due parametri: il nome del file da leggere e una lettera.

La funzione richiesta crea la nuova lista nel modo seguente: legge i nominativi presenti nel file il cui nome riceve come parametro, quindi inserisce nella lista solo i nominativi che non contengono la lettera data (ricevuta come secondo parametro) nei primi 10 caratteri del nominativo.

Esercizio 5

Si consideri la struttura dati che definisce una lista dinamica di nominativi (lunghezza massima di 100 caratteri).

Si scriva quindi la funzione `CreazioneGuidataLista` che restituisce l'indirizzo della testa di una lista di nominativi. La funzione riceve tre parametri: una lista di nominativi da elaborare, un vettore di caratteri e la dimensione del vettore.

La funzione richiesta crea la nuova lista nel modo seguente: i nominativi nella nuova lista vengono presi dalla lista data in base alla lettera iniziale, secondo l'ordine delle lettere riportato nel vettore. Per esempio, se il vettore contiene i tre caratteri 'C', 'Z', 'c' nell'ordine, nella nuova lista ci saranno, nell'ordine, tutti i nominativi della lista data che iniziano con 'C', poi quelli che iniziano con 'Z', quindi quelli che iniziano con 'c'. La nuova lista deve mantenere l'ordine, quindi si deve effettuare un inserimento in coda.

Esercizio 6

Si consideri un programma che gestisce gli orari dei treni di un servizio ferroviario. In particolare, si definisca una lista dinamica che rappresenta le singole fermate di ciascun treno: ogni nodo della lista descrive un numero di treno, il numero progressivo della fermata (la stazione di partenza corrisponde alla fermata numero 1), il nome della stazione, l'ora e i minuti (si usino due campi separati di tipo intero). Nella stessa lista possiamo perciò trovare le fermate di treni diversi.

Dopo aver definito la struttura dati, si scriva in C/C++ la funzione `TreniPiuDiretti`, che riceve come parametro la lista delle fermate, e un numero di treno; la funzione restituisce il numero dei treni con gli stessi capolinea (iniziale e finale) ma che fanno meno fermate intermedie. La funzione restituisce -1 se il treno indicato come parametro non esiste.

Esercizio 7

Si consideri un programma che gestisce gli orari dei treni di un servizio ferroviario. In particolare, si definisca una lista dinamica che rappresenta le singole fermate di ciascun treno: ogni nodo della lista descrive un numero di treno, il numero progressivo della fermata (la stazione di partenza corrisponde alla fermata numero 1), il nome della stazione, l'ora e i minuti (si usino due campi separati di tipo intero). Nella stessa lista possiamo perciò trovare le fermate di treni diversi.

Dopo aver definito la struttura dati, si scriva in C/C++ la funzione `TrenoPiuVeloce`, che riceve come parametro la lista delle fermate, e un numero di treno; la funzione restituisce il numero del treno presente nella lista che effettua lo stesso percorso (prima e ultima fermata uguali) in minor tempo (se il treno ricevuto come parametro è quello più veloce, si restituisce quel numero di treno).
La funzione restituisce -1 se il treno indicato come parametro non esiste.

Esercizio 8

Si consideri il computer di bordo di una automobile MP; tra le varie funzionalità offerte, vi è anche il calcolo del consumo medio. Tale funzionalità calcola il consumo medio a partire da un elenco di rilevazioni effettuate dal computer stesso.

Le rilevazioni sono descritte da una lista dinamica, dove ogni nodo indica i chilometri percorsi dalla precedente rilevazione e i litri consumati.

Dopo aver definito la struttura dati, si scriva in C/C++ la funzione `ConsumoMedio`, che riceve come parametro la lista delle rilevazioni, e restituisce un numero in virgola mobile che indica il consumo medio (in chilometri al litro) calcolato con le rilevazioni presenti nella lista ricevuta come parametro. Se la lista è vuota, la funzione restituisce il valore -1.

Esercizio 9

Si consideri un programma che gestisce i biglietti di un teatro. Ogni biglietto è caratterizzato da un numero progressivo (intero), la data di emissione (stringa di 10 caratteri), dal prezzo del biglietto e dal titolo dello spettacolo (stringa di 50 caratteri). I biglietti sono raccolti in una lista dinamica.

Dopo aver definito la struttura dati, si scriva la funzione `ImportoMassimo`, che riceve come parametri la lista di biglietti, una data (stringa di 10 caratteri) e un numero in virgola mobile denominato soglia; la funzione restituisce l'importo massimo dei prezzi dei biglietti venduti nella data specificata, il cui prezzo è maggiore o uguale della soglia indicata dal terzo parametro. Se in quella data non vi sono biglietti con queste caratteristiche, la funzione restituisce il valore -1.

Esercizio 10

Si consideri un programma che gestisce i biglietti di un teatro. Ogni biglietto è caratterizzato da un numero progressivo (intero), la data di emissione (stringa di 10 caratteri), dal prezzo del biglietto e dal titolo dello spettacolo (stringa di 50 caratteri). I biglietti sono raccolti in una lista dinamica.

Dopo aver definito la struttura dati, si scriva la funzione `ImportoMedio`, che riceve come parametri la lista di biglietti e una data (stringa di 10 caratteri), e restituisce l'importo medio dei prezzi dei biglietti venduti nella data specificata. Se in quella data non vi sono biglietti, la funzione restituisce il valore -1.

Esercizio 11

Si consideri un programma che gestisce percorsi stradali. Un percorso stradale è composto da una sequenza di segmenti; un segmento è costituito da una coordinata iniziale (stringa di 20 caratteri), una coordinata finale (stringa di 20 caratteri) e la lunghezza del segmentino (numero con virgola, indicante i Km del segmento). Un percorso è quindi una lista dinamica di segmenti.

Dopo aver definito la struttura dati, si scriva la funzione `LunghSegmento`, che riceve come parametri un percorso (quindi, l'indirizzo della testa della lista che descrive il percorso) e il numero progressivo di segmento da considerare (il primo segmento del percorso ha progressivo 1, il secondo 2, ecc.): se il segmento esiste nella lista, restituisce la lunghezza del segmento, altrimenti restituisce 0.

Esercizio 12

Il comando di polizia locale vi commissiona la realizzazione della funzione `AggiungiMulta` con la quale si aggiunge un nuovo nodo alla lista passata come parametro. Il nodo deve contenere le informazioni dell'intestatario della targa automobilistica, anch'essa passata come parametro, disponibili nel file della Motorizzazione il cui nome viene passato come terzo parametro alla funzione `AggiungiMulta`.

Dopo aver definito una possibile struttura dati della lista e un esempio di tracciato record del file, implementare la funzione (in C/C++) in modo che riceva i tre parametri descritti, estragga dal file i dati necessari alla creazione di un nuovo nodo da inserire nella lista in base alla targa dell'automobile che ha effettuato l'infrazione e aggiunga un nuovo nodo con tali dati alla lista.

Esercizio 13

Si consideri un programma che gestisce percorsi stradali. Un percorso stradale è composto da una sequenza di segmenti; un segmento è costituito da una coordinata iniziale (stringa di 20 caratteri), una coordinata finale (stringa di 20 caratteri), la lunghezza del segmentino (numero con virgola, indicante i Km del segmento) e la velocità massima (in Km/h) in quel segmento. Un percorso è quindi una lista dinamica di segmenti.

Dopo aver definito la struttura dati, si scriva la funzione `OltreLimiteVelocita`, che riceve come parametri un percorso (quindi, l'indirizzo della testa della lista che descrive il percorso), la distanza percorsa dall'inizio del percorso e la velocità attuale; la funzione restituisce 1 se la velocità attuale è maggiore del limite di velocità del segmento nel quale ci si trova, altrimenti restituisce 0.

Per esempio, se i segmenti del percorso sono 3 e sono lunghi il primo 5 Km, il secondo 10 Km e il terzo 2 Km, e la distanza percorsa è 7 Km, vuol dire che il segmento dove attualmente ci si trova è il secondo, e perciò il suo limite di velocità verrà confrontato con la velocità attuale.

Esercizio 14

Si consideri un programma che gestisce percorsi stradali. Un percorso stradale è composto da una sequenza di segmenti; un segmento è costituito da una coordinata iniziale (stringa di 20 caratteri), una coordinata finale (stringa di 20 caratteri) e la lunghezza del segmentino (numero con virgola, indicante i Km del segmento). Un percorso è quindi una lista dinamica di segmenti.

Dopo aver definito la struttura dati, si scriva la funzione `PercorsoMinimo`, che riceve come parametri due percorsi (quindi, gli indirizzi della testa delle due liste che descrivono i percorsi) e restituisce -1 se il primo percorso è più corto del secondo, restituisce 1 se il secondo percorso è più corto dl primo, 0 se i due percorsi hanno la stessa lunghezza.

Esercizio 15

Implementare la funzione `MediaClientiGrossi`, che riceve il nome del file contenente un elenco di fatture emesse per diversi clienti, il codice numerico di un cliente e un numero minimo di fatture. La funzione restituisce al chiamante il valore medio degli importi delle fatture emesse per il cliente specificato come parametro se il numero di fatture emesse per quel cliente è maggiore o uguale del numero minimo di fatture ricevuto come parametro, altrimenti restituisce zero.

I dati utilizzati da tale funzione si trovano nel file il cui nome è ricevuto come parametro formale e così strutturato:

- numero fattura (6 caratteri)
- codice cliente (6 caratteri)
- data fattura (10 caratteri)
- totale fattura in euro (8 caratteri)

Esercizio 16

Implementare la funzione `TotaleCliente`, che riceve il nome del file contenente un elenco di fatture emesse per diversi clienti, un importo minimo e il codice numerico di un cliente. La funzione restituisce al chiamante il numero totale di fatture emesse per quel cliente di importo superiore all'importo minimo.

I dati utilizzati da tale funzione si trovano nel file il cui nome è ricevuto come parametro formale e così strutturato:

- numero fattura (6 caratteri)
- codice cliente (6 caratteri)
- data fattura (10 caratteri)
- totale fattura in euro (8 caratteri)

Esercizio 17

Il nuovo navigatore satellitare VM.GP necessita della funzione `RemainingRoute()`, che, dato un percorso e la lunghezza percorsa, genera il percorso rimanente. Vediamo nel dettaglio.

Un percorso è descritto da una lista dinamica, dove ogni elemento della lista descrive un segmento di strada da percorrere, con le coordinate iniziali (stringa di 10 caratteri), le coordinate finali (stringa di 10 caratteri), la lunghezza del segmento di strada (numero in virgola mobile indicante i chilometri).

La funzione `RemainingRoute()` riceve due parametri: uno è l'indirizzo della testa della lista che descrive il percorso calcolato, l'altro è la lunghezza già percorsa (numero in virgola mobile indicante i chilometri) di quel percorso. La funzione produce un'altra lista, definita sulla stessa struttura dati e l'indirizzo della cui testa viene restituito dalla funzione, che contiene solo i segmenti del percorso iniziale successivi al segmento raggiunto con i chilometri percorsi (secondo parametro della funzione).

Per esempio, se il percorso calcolato è composto da tre segmenti, il primo di 5 Km, il secondo di 10 Km e il terzo di 2 Km, e la lunghezza percorsa è 7 Km, la funzione produrrà una lista che descrive un percorso costituito solo dal terzo segmento (perché successivo al segmento raggiunto percorrendo 7 Km).
Definire la struttura dati della lista e implementare in C/C++ la funzione richiesta.

Esercizio 18

Il nuovo navigatore satellitare VM.GP necessita della funzione `BackRoute()`, che, dato un percorso e la lunghezza percorsa, genera la parte del percorso fino a quel momento percorsa. Vediamo nel dettaglio.

Un percorso è descritto da una lista dinamica, dove ogni elemento della lista descrive un segmento di strada da percorrere, con le coordinate iniziali (stringa di 10 caratteri), le coordinate finali (stringa di 10 caratteri), la lunghezza del segmento di strada (numero in virgola mobile indicante i chilometri).

La funzione `BackRoute()` riceve due parametri: uno è l'indirizzo della testa della lista che descrive il percorso calcolato, l'altro è la lunghezza già percorsa (numero in virgola mobile indicante i chilometri) di quel percorso. La funzione produce un'altra lista, definita sulla stessa struttura dati e l'indirizzo della cui testa viene restituito dalla funzione, che contiene solo i segmenti del percorso fino a quel momento attraversati, quindi dal primo fino al segmento raggiunto con i chilometri percorsi (secondo parametro della funzione).

Per esempio, se il percorso calcolato è composto da tre segmenti, il primo di 5 Km, il secondo di 10 Km e il terzo di 2 Km, e la lunghezza percorsa è 7 Km, la funzione produrrà una lista che descrive un percorso costituito dal primo e dal secondo segmento.
Definire la struttura dati della lista e implementare in C/C++ la funzione richiesta.

Esercizio 20

Il nuovo navigatore satellitare VM.GP necessita della funzione `LoadRoute()`, che, ricevuto come parametro il nome di un file da aprire che descrive un percorso calcolato precedentemente, restituisca la lista ottenuta caricando le informazioni del percorso. Ogni elemento del percorso sul file è costituito da un segmento di strada da percorrere, con le coordinate iniziali (stringa di 10 caratteri), le coordinate finali (stringa di 10 caratteri), la lunghezza del segmento di strada (numero con virgola di 10 caratteri).

La funzione `LoadRoute()` riceve due parametri: uno è il nome del file da caricare, il secondo è l'indirizzo della testa della lista da caricare. La funzione restituisce 1 se va tutto bene, 0 se qualche cosa non è andato a buon fine.

Definire la struttura dati della lista, in modo che oltre alle informazioni relative ad un segmento così come sono riportate nel file, vi sia anche il totale parziale del percorso, che la funzione deve calcolare per ogni segmento prima di terminare il lavoro.

Implementare in C/C++ la funzione richiesta.