Esame 20220617

Esercizio 3

(1) **Esercizio 3 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Un sistema di gestione delle presenze sul luogo lavorativo assegna ad ogni impiegato un codice (un intero). Ogni impiegato all'ingresso nel luogo di lavoro "timbra il cartellino", e così ad ogni uscita. Il sistema di gestione tiene traccia in due liste differenti le timbrature in ingresso e le timbrature in uscita: ogni volta che un impiegato entra/esce, il sistema aggiunge il codice identificativo in testa alla lista opportuna; uno stesso codice può quindi comparire ripetuto all'interno della rispettiva lista di entrata/uscita. In un mondo perfetto il numero di ingressi sul luogo di lavoro ed il numero di uscite dovrebbe essere sempre uguale, o soddisfare una certa condizione specificata dal datore di lavoro (e.g., numero pari di entrate/uscite). Invece ogni giorno ci sono dei problemi.

Sia data una lista concatenata di interi `list`, terminata con `NULL`, che rappresenta il codice identificativo di un dipendente.

Si scriva una funzione `estrai` che prende come argomenti tre liste concatenate di interi: `dipendenti`, `entrate`, ed `uscite`. Tutte queste liste sono di tipo puntatore al tipo `list` (si veda il file `esercizio3.cpp`).

- La lista di interi `dipendenti` rappresenta i codici identificativi di tutti gli impiegati (**codici non ripetuti**);
- La lista di interi `entrate` contiene i codici dei dipendenti che nella giornata odierna hanno timbrato l'ingresso (ad ogni timbro corrisponde un codice: più timbri di ingresso effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista);
- La lista di interi `uscite` contiene i codici dei dipendenti impiegati che nella giornata odierna hanno timbrato l'uscita (ad ogni timbro corrisponde un codice: più timbri di uscita effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista).

La funzione `estrai(...)` deve **restituire in uscita una lista concatenata di interi** rappresentante i codici corrispondenti alle persone tali per cui il **numero delle entrate è uguale al numero delle uscite**.

Esempio:

```
Lista dipendenti: 63 22 50 93 36 94 16 78 87 84
Lista entrate: 84 94 93 22 78 16 94 93 50 63 87 16 94 93 87
 16 50 22 63 94 36 93 63 87 78 16 94 36 22
Lista uscite: 84 78 93 50 63 84 87 16 50 22 63 78 16 94 36
 50 22 63 84 78 36 93 50 22 63 87 16 94 93 22
Estratti: 36
```

Si scriva inoltre una funzione `delete_list` che deallochi una lista concatenata di interi (terminata con `NULL`).

Note:

- Scaricare il file `esercizio3.cpp`, modificarlo per inserire la dichiarazione e la definizione delle funzioni `estrai` e `delete_list`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream` (la libreria `<string>` è usato solo per conversione `char *` in intero).
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

Information for graders:

(2) **Esercizio 3 v2**

ESSAY | marked out of 10 | penalty 0 | File picker

Un sistema di gestione delle presenze sul luogo lavorativo assegna ad ogni impiegato un codice (un intero). Ogni impiegato all'ingresso nel luogo di lavoro "timbra il cartellino", e così ad ogni uscita. Il sistema di gestione tiene traccia in due liste differenti le timbrature in ingresso e le timbrature in uscita: ogni volta che un impiegato entra/esce, il sistema aggiunge il codice identificativo in testa alla lista opportuna; uno stesso codice può quindi comparire ripetuto all'interno della rispettiva lista di entrata/uscita. In un mondo perfetto il numero di ingressi sul luogo di lavoro ed il numero di uscite dovrebbe essere sempre uguale, o soddisfare una certa condizione specificata dal datore di lavoro (e.g., numero pari di entrate/uscite). Invece ogni giorno ci sono dei problemi.

Sia data una lista concatenata di interi `list`, terminata con `NULL`, che rappresenta il codice identificativo di un dipendente.

Si scriva una funzione `estrai` che prende come argomenti tre liste concatenate di interi: `dipendenti`, `entrate`, ed `uscite`. Tutte queste liste sono di tipo puntatore al tipo `list` (si veda il file `esercizio3.cpp`).

- La lista di interi `dipendenti` rappresenta i codici identificativi di tutti gli impiegati (**codici non ripetuti**);
- La lista di interi `entrate` contiene i codici dei dipendenti che nella giornata odierna hanno timbrato l'ingresso (ad ogni timbro corrisponde un codice: più timbri di ingresso effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista);
- La lista di interi `uscite` contiene i codici dei dipendenti impiegati che nella giornata odierna hanno timbrato l'uscita (ad ogni timbro corrisponde un codice: più timbri di uscita effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista).

La funzione `estrai(...)` deve **restituire in uscita una lista concatenata di interi** rappresentante i codici corrispondenti alle persone tali per cui il **numero delle entrate è diverso dal numero delle uscite**.

Esempio:

```
Lista dipendenti: 63 22 50 93 36 94 16 78 87 84
Lista entrate: 84 94 93 22 78 16 94 93 50 63 87 16 94 93 87
 16 50 22 63 94 36 93 63 87 78 16 94 36 22
Lista uscite: 84 78 93 50 63 84 87 16 50 22 63 78 16 94 36
 50 22 63 84 78 36 93 50 22 63 87 16 94 93 22
Estratti: 84 87 78 16 94 93 50 22 63
```

Si scriva inoltre una funzione `delete_list` che deallochi una lista concatenata di interi (terminata con `NULL`).

Note:

- Scaricare il file `esercizio3.cpp`, modificarlo per inserire la dichiarazione e la definizione delle funzioni `estrai` e `delete_list`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.

- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream` (la libreria `<string>` è usato solo per conversione `char *` in intero).
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

Information for graders:

(3) **Esercizio 3 v3**

ESSAY | marked out of 10 | penalty 0 | File picker

Un sistema di gestione delle presenze sul luogo lavorativo assegna ad ogni impiegato un codice (un intero). Ogni impiegato all'ingresso nel luogo di lavoro "timbra il cartellino", e così ad ogni uscita. Il sistema di gestione tiene traccia in due liste differenti le timbrature in ingresso e le timbrature in uscita: ogni volta che un impiegato entra/esce, il sistema aggiunge il codice identificativo in testa alla lista opportuna; uno stesso codice può quindi comparire ripetuto all'interno della rispettiva lista di entrata/uscita. In un mondo perfetto il numero di ingressi sul luogo di lavoro ed il numero di uscite dovrebbe essere sempre uguale, o soddisfare una certa condizione specificata dal datore di lavoro (e.g., numero pari di entrate/uscite). Invece ogni giorno ci sono dei problemi.

Sia data una lista concatenata di interi `list`, terminata con `NULL`, che rappresenta il codice identificativo di un dipendente.

Si scriva una funzione `estrai` che prende come argomenti tre liste concatenate di interi: `dipendenti`, `entrate`, ed `uscite`. Tutte queste liste sono di tipo puntatore al tipo `list` (si veda il file `esercizio3.cpp`).

- La lista di interi `dipendenti` rappresenta i codici identificativi di tutti gli impiegati (**codici non ripetuti**);
- La lista di interi `entrate` contiene i codici dei dipendenti che nella giornata odierna hanno timbrato l'ingresso (ad ogni timbro corrisponde un codice: più timbri di ingresso effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista);
- La lista di interi `uscite` contiene i codici dei dipendenti impiegati che nella giornata odierna hanno timbrato l'uscita (ad ogni timbro corrisponde un codice: più timbri di uscita effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista).

La funzione `estrai(...)` deve **restituire in uscita una lista concatenata di interi** rappresentante i codici corrispondenti alle persone tali per cui la **differenza tra le entrate e le uscite è un numero pari**.

Esempio:

```
Lista dipendenti: 63 22 50 93 36 94 16 78 87 84
Lista entrate: 84 94 93 22 78 16 94 93 50 63 87 16 94 93 87
 16 50 22 63 94 36 93 63 87 78 16 94 36 22
Lista uscite: 84 78 93 50 63 84 87 16 50 22 63 78 16 94 36
 50 22 63 84 78 36 93 50 22 63 87 16 94 93 22
Estratti: 84 36 50
```

Si scriva inoltre una funzione `delete_list` che deallochi una lista concatenata di interi (terminata con `NULL`).

Note:

- Scaricare il file `esercizio3.cpp`, modificarlo per inserire la dichiarazione e la definizione delle funzioni `estrai` e `delete_list`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.

- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream` (la libreria `<string>` è usato solo per conversione `char *` in intero).
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

Information for graders:

(4) **Esercizio 3 v4**

ESSAY | marked out of 10 | penalty 0 | File picker

Un sistema di gestione delle presenze sul luogo lavorativo assegna ad ogni impiegato un codice (un intero). Ogni impiegato all'ingresso nel luogo di lavoro "timbra il cartellino", e così ad ogni uscita. Il sistema di gestione tiene traccia in due liste differenti le timbrature in ingresso e le timbrature in uscita: ogni volta che un impiegato entra/esce, il sistema aggiunge il codice identificativo in testa alla lista opportuna; uno stesso codice può quindi comparire ripetuto all'interno della rispettiva lista di entrata/uscita. In un mondo perfetto il numero di ingressi sul luogo di lavoro ed il numero di uscite dovrebbe essere sempre uguale, o soddisfare una certa condizione specificata dal datore di lavoro (e.g., numero pari di entrate/uscite). Invece ogni giorno ci sono dei problemi.

Sia data una lista concatenata di interi `list`, terminata con `NULL`, che rappresenta il codice identificativo di un dipendente.

Si scriva una funzione `estrai` che prende come argomenti tre liste concatenate di interi: `dipendenti`, `entrate`, ed `uscite`. Tutte queste liste sono di tipo puntatore al tipo `list` (si veda il file `esercizio3.cpp`).

- La lista di interi `dipendenti` rappresenta i codici identificativi di tutti gli impiegati (**codici non ripetuti**);
- La lista di interi `entrate` contiene i codici dei dipendenti che nella giornata odierna hanno timbrato l'ingresso (ad ogni timbro corrisponde un codice: più timbri di ingresso effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista);
- La lista di interi `uscite` contiene i codici dei dipendenti impiegati che nella giornata odierna hanno timbrato l'uscita (ad ogni timbro corrisponde un codice: più timbri di uscita effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista).

La funzione `estrai(...)` deve **restituire in uscita una lista concatenata di interi** rappresentante i codici corrispondenti alle persone tali per cui la **differenza tra le entrate e le uscite è un numero dispari**.

Esempio:

```
Lista dipendenti: 63 22 50 93 36 94 16 78 87 84
Lista entrate: 84 94 93 22 78 16 94 93 50 63 87 16 94 93 87
 16 50 22 63 94 36 93 63 87 78 16 94 36 22
Lista uscite: 84 78 93 50 63 84 87 16 50 22 63 78 16 94 36
 50 22 63 84 78 36 93 50 22 63 87 16 94 93 22
Estratti: 87 78 16 94 93 22 63
```

Si scriva inoltre una funzione `delete_list` che deallochi una lista concatenata di interi (terminata con `NULL`).

Note:

- Scaricare il file `esercizio3.cpp`, modificarlo per inserire la dichiarazione e la definizione delle funzioni `estrai` e `delete_list`, e caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito.

- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream` (la libreria `<string>` è usato solo per conversione `char *` in intero).
- Si ricorda che, l'esempio di esecuzione è puramente indicativo, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.

Information for graders:

Total of marks: 40