S294424

Relazione Esame

Strutture dati:

```c
typedef struct activity_wrapper_s * Activity_Wrapper;  // Quasi ADT
typedef struct {
    int id; //corrispondenza id pos per sicurezza
    int diff;
} Activity;
struct activity_wrapper_s {
    int N;
    Activity * vett;
};

typedef struct person_wrapper_s * Person_Wrapper;  // ADT 1 classe
typedef struct {
    int id; // corrispondenza id pos per sicurezza
    int exp;
    int available;
} Person; // ADT 1 classe

struct person_wrapper_s {
    int N;
    Person * vett;
};
```

Si è deciso di aggiungere sia a **Person**, sia a **Activity**, il campo id con corrispondenza indice vettore per soddisfare alcune necessità nella risoluzione del problema e maggior chiarezza.
Durante lo svolgimento del compito, **Activity** e **Activity_Wrapper** sono stati dichiarati come ADT di 1 classe, ma successivamente ho ritenuto necessario farli diventare quasi ADT. Infatti, le funzioni di verifica e generazione soluzione sono in **person.c**, il quale deve accendere ai campi di queste due struct.
Le due strutture wrapper sono state introdotte per rendere più agevole la gestione dei vettori attività e persone.
Inoltre, si è ritenuto opportuno salvare le sinergie in una semplice matrice di interi dove id riga e colonna indicano la coppia di persone. Nella soluzione proposta durante il compito viene dichiarata erratamente la matrice. Nella versione corretta quest'ultima viene creata dinamicamente.
I vari caricamenti e liberazione memoria vengono effettuate con le funzioni `Person_Wrapper PERSONLoad(FILE * fin), void PERSONFree(Person_Wrapper p_wrapper, int ** mat), int ** PERSONBuildMat(FILE * fin, Person_Wrapper p_wrapper), Activity_Wrapper ACTIVITYLoad(FILE * fin), void ACTIVITYFree(Activity_Wrapper a_wrapper).`

Risoluzione problema verifica soluzione proposta:

Per soddisfare questo requisito, si è deciso di creare una funzione

```
int verify(int *person, int N_person, int id_activity, Person_Wrapper p_wrapper, Activity_Wrapper a_wrapper, int **mat)
```

Quest'ultima riceve in `person` un vettore contenente gli id (o indice nel vett di **p_wrapper**) delle persone assegnate all'incarico. La funzione ritorna l'esperienza dell'attività (se i criteri sono soddisfatti, altrimenti -1), verifica i requisiti esperienza minima e che il personale non sia assegnato già ad un'altra attività con un controllo del campo `int available` all'interno di **Person**. Per quanto concerne il punteggio derivante dalle sinergie, è stata creata una funzione dedicate chiamata:

```
int searchSinergie(int * person, int N_person, int **mat)
```

Quest'ultima, sfruttando una funzione di generazione di **combinazioni semplici con k=2** tra il personale predisposto all'attività, calcola il punteggio derivante dalle sinergie e lo ritorna alla funzione `int verify.`

Differenze particolari rispetto alla soluzione proposta durante il compito non ce ne sono.
Gli errori principali sono l'aver dimenticato la variabile `int start` per il calcolo delle combinazioni e la restituzione del calcolo del punteggio dalla funzione `int verify`.

Generazione soluzione ottima:

Per quanto riguarda questo punto, durante il compito si era pensata una soluzione basata sul partizionamento del personale con la funzione **ER** con k numero di attività, e successivamente applicare disposizioni semplici su ciascuna soluzione dell'algoritmo **ER** per variare la posizione di ogni partizionamento nel vettore soluzione. Successivamente, mi sono reso conto che quello che cercavo di fare era solamente una disposizione semplice del personale.
La soluzione corretta implementa disposizioni semplici. il vettore sol ha come dimensione il numero delle persone e il contenuto di ogni cella varia da 0 a n_attivita-1 (indice vettore indica id persona, il contenuto l'indice dell'attività). In questo modo si ottiene lo stesso comportamento della soluzione proposta durante il compito non completata in tempo e più facilmente.
Successivamente ogni partizione associata ad ogni attività (se esiste) viene verificata con la `int verify` e se tutte le partizioni soddisfano i requisiti, la soluzione trovata viene confrontata con la migliore soluzione corrente.

Altri errori di minor rilevanza sono riportati come commenti con al termine la sigla ERRORE tutti all'interno del progetto.
Ulteriori informazioni sulla risoluzione sono riportate come commento all'interno del progetto con anche esempi per le variabili più complesse utilizzate