Fondamenti di Informatica T
Prova Pratica – 15 Febbraio 2021
Compito

Prima di cominciare: *si scarichino i file di testo dallo* ***StartKit.zip*** *dopo aver opportunamente unzippato la cartella e i file contenuti in essa.*
Avvertenze per la consegna:
1) nominare TUTTI i file sorgenti ***come richiesto nel testo del compito****, apporre all'inizio di* ***ogni file*** *sorgente un commento contenente i propri dati (****cognome, nome, numero di matricola****).*

2) Al termine, **consegnare tutti i file sorgenti** (i due file .c e il file .h) **ed i file di testo contenuti nello StartKit**.
NON CONSEGNARE CARTELLE, PROGETTI O .ZIP
3) Rispettare le specifiche del testo, in particolare inserire le funzioni con i nomi, i parametri di ingresso e i parametri di uscita **esattamente come indicato nel testo**. Chi non rispetta le specifiche sarà opportunamente penalizzato.

4) ***SARANNO CORRETTI SOLO gli elaborati SENZA errori di compilazione****.*

La società *Costa Armonia* si occupa di trasporto merci per via marittima, con base a Livorno. Per questo la compagnia ha sviluppato un componente software per la gestione delle partenze delle navi e dei ritardi. Gestendo infatti solo carico merci e non passeggeri, la società può permettersi sia di ritardare che di anticipare le partenze dei suoi mezzi.
La società dispone di alcuni file con salvati diversi dati, il primo file contiene informazioni sulle navi ("navi.txt") ed è organizzato nel seguente modo:

codice_nave tipo

Dove **codice_nave** è una stringa di 5 caratteri (terminatore escluso) che identifica univocamente la nave, **tipo** invece è un singolo carattere che indica se la nave è del tipo veloce 'V' o lento 'L'.

Il file "Partenze.txt" contiene la lista delle partenze ed è formato nel seguente modo:

id_viaggio partenza arrivo codice_nave orario ritardo

Dove **id_viaggio** è un intero che identifica in modo univoco il viaggio, **partenza** e **arrivo** sono due stringhe di 40 caratteri (senza spazi) contenenti rispettivamente il nome della città di partenza e di arrivo, **codice_nave** è una sequenza di 5 caratteri che identifica la nave, **orario** è un intero contenente l'orario di partenza in secondi dalla mezzanotte (es: se l'orario di partenza è 11:55:12 allora il campo conterrà il valore 11*60*60+55*60+12=42912), **ritardo** è un intero contenente il ritardo in secondi.

Fondamenti di Informatica T
Prova Pratica – 15 Febbraio 2021
Compito

Esercizio 1 - Lettura da file (compagnia_navi.h/ compagnia_navi.c)

Si definisca un'opportuna struttura dati **nave** al fine di rappresentare e tenere traccia delle navi presenti.
Il candidato realizzi poi la funzione:

nave * leggi_navi (char *nomefile, int *num_n)

che, ricevuto come parametro di ingresso il nome di un file, provveda ad aprirlo e ne legga il contenuto, ovvero la lista delle navi. Non è noto a priori quante navi siano presenti nel file, quindi sarà necessario allocare memoria dinamicamente nella dimensione opportuna. Tale funzione deve restituire la dimensione logica dell'array tramite l'intero **num_n.**
Si realizzi inoltre una procedura:

void stampa_nave (nave n)

Che stampi a video (in maniera leggibile – usare spazi, separatori, fine linea, etc.) le informazioni relative ad una singola nave presa come ingresso dalla procedura.

Si definisca la struttura **partenza** in modo da contenere tutte le informazioni relative alle tratte. Si crei una funzione:

partenza * leggi_partenze(char *nomefile, int *num_p)

Si implementi la funzione in modo tale che, dato il nome del file contenente i dati delle partenze, crei dinamicamente l'array di strutture **partenza** della dimensione esatta e restituisca, oltre al vettore, il numero di partenze lette **num_p**.
Si realizzi la procedura:

void stampa_partenza(partenza p)

Che stampa tutte le informazioni relative ad una partenza passata in ingresso.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, sfruttando il file di testo "*navi.txt*" e "*partenze.txt*" (presenti nello **StartKit.zip**). Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

Esercizio 2 – Nuova partenza(compagnia_navi.h/ compagnia_navi.c)

Si definisca la funzione

int trova_nave(char *codice, nave *n, int num_n)

che, preso in ingresso una stringa contenente il codice di una nave, cerca all'interno del vettore delle navi **n**, se questa è presente; **num_n** contiene la dimensione logica dell'array di navi. La funzione dovrà restituire 1 se la nave è stata trovata, 0 altrimenti.

Si definisca la funzione

int nuova_partenza(FILE *fp, nave *n, int num_n)

che prende in ingresso il puntatore ad un file già aperto in modalità "append", un vettore **n** di strutture dati **nave** e la relativa dimensione logica; tale funzione deve inserire all'interno del file delle partenze, una nuova partenza (inserita da tastiera), avendo cura di verificare che il codice della nave della nuova partenza sia presente nel vettore delle navi

(si utilizzi a tale scopo la funzione **trova_nave**).
La funzione deve restituire 1 se l'inserimento è andato a buon fine, -1 se la nave non è stata trovata, 0 in caso di altri errori.

Si realizzino nel main le opportune istruzioni per verificare il corretto funzionamento della funzione, scrivendo la nuova partenza nel file "*partenze.txt*".

Esercizio 3 – Ordinamento partenze (compagnia_navi.h/ compagnia_navi.c)

Si realizzi la procedura:

void ordina_partenze(partenza *p, int num_p)

Che preso in ingresso un vettore **p** di strutture dati **partenza**, con la sua dimensione logica, si occupi di ordinare le partenze sulla base dell'orario di partenza effettivo (**orario** + **ritardo**) in ordine crescente.

Esercizio 4 – Main (main.c)

Il candidato realizzi un programma in grado di:

1) Leggere navi e partenze dai rispettivi file creando due array;
2) Mostrare a video tutti le navi e le partenze;
3) Inserire una nuova partenza;
4) Ordinare le partenze e stampare a video le partenze ordinate;
5) Deallocare (al termine del programma) tutte le strutture allocate dinamicamente.