Laboratorio di Informatica L-A
Prova d'Esame 4 – 27 Giugno 2008

Prima di cominciare: loggarsi sul sito http://esamix.labx e scaricare il file **`StartKit4.zip`** contenente i file necessari.
Avvertenze per la consegna: nominare 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**). Al termine, **consegnare tutti i file sorgenti** ed i file contenuti nello StartKit.
Rispettare le specifiche, in particolare inserire le funzioni nei file specificati fra parentesi dopo il nome della funzione. Chi non rispetta le specifiche sarà opportunamente penalizzato. **NON SARANNO CORRETTI** gli elaborati che presenteranno un numero "non affrontabile" di errori di compilazione.
Consiglio: per verificare l'assenza di *warnings*, effettuare di tanto in tanto un *Rebuild All*.

Un remoto paese dell'Europa dell'Est ha descritto tutte le tratte della propria rete nazionale nel file Trains.txt (contenuto nello *start kit*) dove ogni riga descrive un treno e comprende nell'ordine:

- il numero del treno (un intero)
- un carattere ';' di separazione
- il nome del treno (una stringa)
- un carattere ';' di separazione
- una lista di stazioni con relativi orari di arrivo e partenza rappresentati come segue:
  - il nome della stazione
  - un carattere ';' di separazione
  - l'orario di arrivo nella stazione nel formato hh.mm (es. 07.05)
  - un carattere '/'
  - l'orario di partenza dalla stazione nel formato hh.mm
  - un carattere ';' di separazione

  la lista delle stazioni è terminata dalla stringa "END".

Si desidera realizzare un'applicazione che consenta di leggere e memorizzare il file sopra menzionato e di effettuare ricerche di viaggio date una stazione di partenza ed una di arrivo.

Esercizio 1 – Definizione della strutture dati (time.h/time.c, trainStop.h/trainStop.c, train.h/train.c)

Definire un tipo di dato di nome **`Time`** (**`time.h/time.c`**) basato su una struttura ed atto a contenere i dati relativi ad un'ora del giorno (o ad una differenza di ore del giorno) – la struttura di base deve contenere ore e minuti (interi). Per tale tipo di dato devono essere scritte le funzioni **`timeFromString`**, **`difference`** e **`timeToString`** (**`time.h/time.c`**). La funzione **`timeFromString`** prende in ingresso una stringa e restituire un puntatore a **`Time`**; nel caso in cui la stringa in ingresso valga **`"--.--"`**, la funzione deve restituire **`NULL`**, altrimenti deve tentare di convertire un'ora nel formato "hh.mm" e restituirla in un puntatore ad una struttura **`Time`** opportunamente inizializzata. La funzione **`difference`** prende in ingresso due **`Time`** e restituisce un **`Time`** contenente la differenza fra i due in ingresso. La funzione **`timeToString`** prende in ingresso un puntatore a **`Time`** e restituisce una stringa nel formato "hh.mm" che lo rappresenta (se il puntatore è nullo, restituire la stringa **`"--.--"`**).
Definire un tipo di dato di nome **`TrainStop`** (**`trainStop.h/trainStop.c`**) basato su una struttura ed atto a contenere i dati relativi ad una fermata – la struttura di base deve contenere il nome della stazione (una stringa di al più 50 caratteri), un puntatore a **`Time`** che rappresenta l'orario di arrivo nella stazione (può essere **`NULL`** per i treni che hanno origine in quella stazione), un puntatore a **`Time`** che rappresenta l'orario di partenza dalla stazione (può essere **`NULL`** per i treni di cui la stazione è capolinea).
Definire un tipo di dato di nome **`Train`** (**`train.h/train.c`**) basato su una struttura ed atto a contenere i dati relativi ad un treno – la struttura di base deve contenere il codice del treno (un intero), il nome del treno (una stringa di al più 40 caratteri), una *linked list* di **`TrainStop`**. Per la lista si utilizzi l'ADT **`TrainStopList`** contenuto nello *start kit*.
Attenzione: nel file **`TrainStop.h`**, il tipo **`TrainStop`** è attualmente definito come intero (per evitare errori di compiazione); eliminare la definizione corrente e sostituirla con la propria.

Laboratorio di Informatica L-A
Prova d'Esame 4 – 27 Giugno 2008

Nel **main**, si verifichi il funzionamento del codice scritto con particolare riferimento alle funzioni relative al tipo di dato **Time**. Al termine del test commentare il codice senza eliminarlo.

Esercizio 2 – Lettura del file (train.h/train.c)

Definire una funzione **readTrainsFromTxt** (**train.h/train.c**) che prenda in ingresso un puntatore a file e restituisca sia un *array* di **Train** (allocato dinamicamente di dimensioni "esatte") sia la dimensione dell'*array* stesso. Si suggerisce (non è obbligatorio!) di suddividere il lavoro di lettura in più sotto-funzioni: **readTrainsFromTxt** può usare una funzione di lettura di un singolo treno che può usare una funzione di lettura di una lista di fermate che può usare la funzione di lettura di una singola fermata… Poiché la lista delle stazioni va costruita in avanti e non all'indietro (**cons**), progettare un'opportuna funzione **addToList** che consenta di aggiungere una nuova fermata in fondo alla lista.
Definire una funzione **printTrain** (**train.h/train.c**) che prenda in ingresso un **Train** e lo stampi a video completo del dettaglio delle fermate; per ottenere una rappresentazione in stringa delle strutture **Time** contenute nelle fermate, utilizzare la funzione **timeToString** precedentemente progettata.
Definire una funzione **printTrains** (**train.h/train.c**) che prenda in ingresso un *array* di **Train** e la sua dimensione logica e stampi a video il contenuto dell'*array*; tale funzione deve usare la funzione **printTrain** appena definita.
Nel **main**, si verifichi il funzionamento del codice scritto inserendo le istruzioni necessarie per la lettura del file **Trains.txt** e stampando ciò che è stato letto. Al termine del test commentare il codice che effettua la stampa senza eliminarlo.

Esercizio 3 – Ricerca di una soluzione di viaggio (train.h/train.c)

Definire una funzione di nome **extractStopListFromTrain** che prenda in ingresso un **Train**, una stazione di partenza (una stringa) e una stazione di arrivo (una stringa) e restituisca una nuova **TrainStopList** che contenga le sole fermate dalla stazione di partenza alla stazione di arrivo; nel caso in cui il treno non fermi in una delle due o in entrambe le stazioni di fermata ricevute in ingresso, occorre restituire la lista vuota. Definire una funzione **tripTime** che, presa in ingresso una **TrainStopList,** calcoli il tempo di percorrenza fra la prima e l'ultima stazione della lista e lo restituisca sotto forma di **Time**.
Definire una funzione **tripPrice** che, presa in ingresso una **TrainStopList,** calcoli il costo del viaggio addebitando 0,40€ per ogni 10 minuti (o frazione) di percorrenza (un viaggio di 15 minuti costa pertanto 0,80€). Nel main inserire il codice necessario per:

- richiedere all'utente l'inserimento delle stazioni di partenza e di arrivo
- effettuare una ricerca su tutto l'*array* di **Train** e per ogni **Train** per cui la funzione **extractStopListFromTrain** restituisce una lista non vuota
    - calcolare il tempo di percorrenza (**tripTime**) e il costo (**tripPrice**) del viaggio
    - stampare il **Train** (**printTrain**)
    - stampare il tempo di percorrenza (usare la funzione **timeToString** per avere una rappresentazione in stringa del **Time**)
    - stampare il costo del viaggio