Laboratorio di Informatica L-A
Prova d'Esame 6 – 18 Settembre 2008

Prima di cominciare: loggarsi sul sito http://esamix.labx e scaricare il file **`StartKit6.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'agenzia di navigazione desidera informatizzare la gestione dei viaggi delle proprie navi merci. In un file di testo (Ships.txt) sono programmati i viaggi e relative le navi: i dati sono memorizzati in una riga del file di cui sopra che comprende, nell'ordine:

- il nome della nave (una stringa di al più 16 caratteri) + un carattere ';' di separazione
- la capacità della nave in tonnellate (un intero) + un carattere ';' di separazione
- il nome del porto di partenza (una stringa di al più 16 caratteri) + un carattere ';' di separazione
- la data di partenza nel formato gg/mm/aaaa + un carattere 'T' di separazione
- l'orario di partenza (ora locale del porto di partenza con offset rispetto al GMT – vedere file Ships.txt per esempi) + un carattere ';' di separazione
- il nome del porto di arrivo (una stringa di al più 16 caratteri) + un carattere ';' di separazione
- la data di arrivo nel formato gg/mm/aaaa + un carattere 'T' di separazione
- l'orario di arrivo (ora locale del porto di arrivo con offset rispetto al GMT – vedere file Ships.txt per esempi)

Ogni nave è in grado di trasportare una certa quantità di merce dal porto di partenza al porto di destinazione. La merce da trasportare è memorizzata in un file di testo (Goods.txt) in cui ogni riga rappresenta un bene e comprente, nell'ordine, le informazioni che seguono:

- il nome del bene (una stringa di al più 20 caratteri) + un carattere ';' di separazione
- il peso della merce in tonnellate (un intero)
- il nome del porto di partenza (una stringa di al più 16 caratteri) + un carattere ';' di separazione
- il nome del porto di arrivo (una stringa di al più 16 caratteri) + un carattere ';' di separazione

Si desidera realizzare un'applicazione che consenta di leggere e memorizzare il file dei viaggi sopra menzionato e di effettuare la stampa dei viaggi stessi con il calcolo della relativa durata (esercizi 1 e 2). Successivamente si deve estendere il sistema consentendo all'utente di caricare il file dei beni e di computare quale merce vada in quale nave a seconda della partenza, della destinazione, del peso della merce e della capacità della nave (esercizio 3).

Esercizio 1 –DateTime e TimeSpan (time.h/time.c)

Definire un tipo di dato di nome **`DateTime`** (**`time.h/time.c`**) basato su una struttura ed atto a contenere i dati relativi ad una data ed un'ora del giorno comprensiva di offset relativo al GMT – la struttura di base deve contenere giorno, mese, anno, ore e minuti (interi) e l'offset (intero che può essere positivo o negativo) rispetto al GMT. Per tale tipo di dato devono essere scritte le funzioni **`fillDateTime`**, **`dateTimeDifference`** e **`dateTimeToString`** (**`time.h/time.c`**). La funzione **`fillDateTime`** prende in ingresso sei interi (nell'ordine giorno, mese, anno, ore, minuti e offset) e un **`DateTime`** per riferimento e restituisce un valore booleano che sarà falso nel caso di conversione fallita (verificare che giorno, mese, ore, minuti e offset abbiano valori sensati; per il giorno avvalersi della funzione **`isLeapYear`** contenuta nello start kit che, dato un anno, indica se sia o meno bisestile), vero altrimenti; nel caso in cui la conversione abbia avuto successo, il risultato deve essere inserito nella struttura **`DateTime`** ricevuta per riferimento. La funzione **`dateTimeDifference`** prende in ingresso due **`DateTime`** e restituisce un **`TimeSpan`** contenente la differenza fra i due in ingresso in termini di giorni, ore, minuti – si ricorda che è necessario tenere conto sia dei fusi orari, sia del numero di giorni presenti in un certo mese, sia del fatto che

gli anni in gioco siano o meno bisestili; per semplicità, non possono esistere viaggi che coinvolgano il capodanno e che attraversino la linea del cambio di data – per la codifica, si tenga eventualmente conto del suggerimento in calce al testo dell'esercizio. La struttura **`TimeSpan`** (da definire) contiene solamente giorni, ore e minuti (interi) poiché rappresenta un lasso di tempo. La funzione **`dateTimeToString`** prende in ingresso un **`DateTime`** e restituisce una stringa nel formato "gg/mm/aaaaThh.mm+/-fuso" che lo rappresenta. Progettare, infine, la funzione **`timeSpanToString`** che prende in ingresso un **`TimeSpan`** e restituisce una stringa nel formato "ggG hhH mmM" che lo rappresenta (es: 10G 20H 32M).

Nel **`main`**, si verifichi il funzionamento del codice scritto. Al termine del test commentare il codice senza eliminarlo.

Suggerimento: per calcolare la differenza fra le date si calcoli prima il numero di giorni dall'inizio dell'anno relativi ad entrambe le date, poi si effettui la sottrazione ed infine si tenga conto di un eventuale riporto derivato dalla differenza delle ore.

Esercizio 2 – Lettura del file dei viaggi (ship.h/ship.c)

Definire un tipo di dato **`Ship`** basato su una struttura in modo che possa ospitare i dati contenuti nel file Ships.txt come specificato nell'introduzione. Definire una funzione **`readShipsFromTxt`** (**`ship.h/ship.c`**) che prenda in ingresso un puntatore a file e restituisca una *linked list* di strutture **`Ship`** (eventualmente utilizzando l'ADT lista). Si suggerisce (non è obbligatorio!) di suddividere il lavoro di lettura in più sotto-funzioni: **`readShipsFromTxt`** può usare una funzione di lettura di un singolo viaggio…
Definire una funzione **`printShip`** (**`ship.h/ship.c`**) che prenda in ingresso uno **`Ship`** e lo stampi a video completo del dato relativo alla durata del viaggio (usare la funzione **`dateTimeDifference`** per il calcolo e la funzione **`timeSpanToString`** per ottenere una rappresentazione in stringa del **`TimeSpan`** ottenuto; per avere una rappresentazione in stringa delle strutture **`DateTime`** contenute nelle fermate, utilizzare la funzione **`dateTimeToString`**).
Definire una funzione **`printShips`** (**`ship.h/ship.c`**) che prenda in ingresso una lista di **`Ship`** e ne stampi a video il contenuto; tale funzione deve usare la funzione **`printShip`** appena definita.

Nel **`main`**, si verifichi il funzionamento del codice scritto inserendo le istruzioni necessarie per la lettura del file **`Ships.txt`** e stampando ciò che è stato letto. Al termine del test commentare il codice che effettua la stampa senza eliminarlo.

Esercizio 3 – Lettura del file dei beni e riempimento delle navi (goods.h/goods.c, ship.h/ship.c)

Definire un tipo di dato **`Good`** basato su una struttura in grado di contenere i dati relativi ad un bene come specificato nell'introduzione. Definire una funzione di nome **`readGoodsFromTxt`** che prenda in ingresso un puntatore a file e che restituisca come valore di ritorno un *array* di **`Good`** risultato della lettura del file, e per riferimento la dimensione fisica dell'*array* stesso (l'*array* deve essere allocato di dimensione esatta rispetto al numero di beni presenti nel file). Definire una funzione **`printGoods`** (a solo scopo di *debug*) che prenda in ingresso un *array* di **`Good`** e la relativa dimensione logica e stampi tutti i beni contenuti. Definire, infine, una funzione di nome **`printShipsWithGoods`** che data una lista di **`Ship`** ed un array di **`Good`** (e la relativa dimensione logica) stampi a video per ogni **`Ship`** contenuto nella lista tutti i **`Good`** che possono far parte del viaggio – si tenti di riempire una nave con i soli beni la cui località di partenza coincida con il porto di partenza della nave e la cui località di arrivo coincida con il porto di arrivo e, per semplicità, considerare i beni nell'ordine in cui compaiono nell'array fermandosi quando un bene supera la capacità della nave.

Nel **`main`** si utilizzi la **`readGoodsFromTxt`** per caricare i dati dal file Goods.txt, si stampi la lista dei beni tramite la funzione **`printGoods`** e si utilizzi, infine, la funzione **`printShipsWithGoods`** per stampare i beni contenuti nelle varie navi.