13 settembre – Fondamenti di Informatica – Seconda Parte – 120 minuti
COMPITO B – Esercizi di Programmazione (23 punti)

Esercizio B1 (7 punti): Stringhe

Realizzare un'applicazione ***MaiuscoleInFondo*** definita come segue.

(6pt) L'applicazione contiene una funzione ***maiuscoleInFondo*** che riceve come parametro una stringa e la modifica riordinando i caratteri della stringa così che tutti i caratteri alfabetici maiuscoli compaiano dopo ciascun carattere che non è alfabetico maiuscolo (con l'eccezione del carattere di fine stringa, che deve rimanere l'ultimo carattere della stringa). L'ordine reciproco dei caratteri alfabetici maiuscoli nella stringa modificata può essere uno qualsiasi, così come l'ordine reciproco dei caratteri che non sono alfabetici maiuscoli. Ad esempio, se **s = "abABcCDEdef"**, allora **"abfecdDECBA"** rappresenta una possibile modifica corretta della stringa.

(1pt) L'applicazione contiene una funzione ***main*** che chiede all'utente di inserire una stringa da tastiera ed utilizzando la funzione *fgets* memorizza la stringa introdotta dall'utente all'interno di un array di 50 caratteri. Dopo aver rimosso il carattere **'\n'** dalla stringa, la funzione *main* invoca la funzione *maiuscoleInFondo* fornendole come parametro la stringa letta; la funzione *main* stampa quindi un messaggio che mostra all'utente la stringa così come è stata modificata dalla funzione *maiuscoleInFondo*.

Esercizio B2 (7 punti): Ricorsione

Realizzare un'applicazione ***NumeroCoppieUguali*** definita come segue.

(4pt) L'applicazione contiene una funzione **ricorsiva** ***numeroCoppieUguali*** che conta, all'interno di un array di interi ricevuto come parametro, quante coppie di interi adiacenti uguali sono presenti nell'array. Ad esempio, l'istanza **[1, 1, 2, 3, 3, 3, 3]** del problema ha soluzione **4**, in quanto fra le coppie di interi adiacenti nell'array (ovvero **[1, 1]**, **[1, 2]**, **[2, 3]**, **[3, 3]**, **[3, 3]**, **[3, 3]**) ce ne sono **4** composte da interi fra loro uguali.

La funzione ***numeroCoppieUguali*** ha due parametri: l'array e la sua lunghezza. Nel caso in cui si desideri utilizzare un terzo parametro per realizzare la ricorsione, deve essere definita un'ulteriore funzione ***numeroCoppieUgualiRic*** che ha tre parametri e che realizza la ricorsione. In tal caso la funzione ***numeroCoppieUguali*** invoca la funzione ***numeroCoppieUgualiRic*** (fornendole opportuni parametri) per calcolare il risultato.

(0.5pt) In un commento che precede la funzione ***numeroCoppieUguali*** descrivere il tipo di problema (accumulazione/conteggio/verifica esistenziale/verifica universale/minimo-massimo/ricerca) che è risolto dall'algoritmo implementato dalla funzione ***numeroCoppieUguali***.

(1pt) In un commento che precede la funzione ***numeroCoppieUguali*** descrivere la specifica della funzione ***numeroCoppieUguali*** (espressa come Input-Precondizione-Output-Postcondizione).

(1.5pt) L'applicazione contiene una funzione ***main*** che gestisce l'interazione con l'utente. La funzione chiede all'utente la lunghezza della sequenza che vuole inserire. Dopo aver letto e memorizzato tale sequenza in un array, viene stampato un messaggio che indica quante coppie di interi adiacenti uguali sono presenti nell'array. La funzione termina salvando il contenuto dell'array su di un file. Il file si deve chiamare interi.dat (nel caso sia binario) oppure interi.txt (nel caso in cui sia testuale). La scelta se utilizzare un file binario oppure un file di testo è lasciata allo studente.

La funzione main è già stata parzialmente implementata in un file ***ricorsione.c*** (ad eccezione dei messaggi finali e del salvataggio dei dati) il cui codice può essere scaricato e copiato da moodle.

Esercizio B3 (9 punti): Liste

Realizzare un'applicazione ***Pizze*** per gestire il menù di una pizzeria.

(1pt) Definire due strutture, una per rappresentare un elemento della lista ed una per rappresentare una pizza che fa parte del menù. In particolare, l'applicazione deve gestire ciascuna pizza come una struttura con tre campi. Il primo campo è una stringa che indica il nome della pizza (ad esempio **"margherita"**), il secondo campo è un valore reale che indica il prezzo della pizza, ed il terzo campo è un carattere **'y'** o **'n'** che indica se la pizza è vegetariana.

(1.5pt) Definire una funzione che visualizza il menù di pizze, stampando per ciascuna pizza il nome, il prezzo e se la pizza è vegetariana oppure no.

(2pt) Definire una funzione che inserisce una pizza in testa al menù.

(4.5pt) Definire una funzione che cancella dal menù la prima pizza vegetariana. Se il menù non contiene alcuna pizza vegetariana, viene stampato un messaggio che informa l'utente di questo fatto.

Definire una funzione main che crea la lista e gestisce l'interazione con l'utente. La funzione main è già parzialmente implementata in un file **listeB.c** che può essere scaricato da Moodle.