Esercitazione di laboratorio n. 3

Esercizio n.1: **Individuazione di regioni**
Competenze: lettura/scrittura di file, manipolazioni di matrici statiche; puntatori e passaggio di parametri per riferimento (Puntatori e strutture dati dinamiche: 1.4)
Categoria: problemi di verifica e selezione (Dal problema al programma: 4.5)

Si riveda l'esercizio 1 del Lab02 apportandovi le seguenti modifiche:

- supponendo di avere dichiarato una matrice di interi `M` e di aver definito `MAXR` come `50` si acquisisca la matrice mediante una funzione (`leggiMatrice`) che ne ritorna il numero di righe e di colonne effettivamente usati, come parametri "by reference" (o meglio, con puntatori by value). La funzione deve poter essere chiamata con un'istruzione del tipo:

```
leggiMatrice(M,MAXR,&nr,&nc);
```

- per effettuare il riconoscimento delle regioni si utilizzi una funzione `riconosciRegione` che, data una casella della matrice, determini se si tratti o meno di estremo superiore sinistro di una regione, ritornandone "by reference" (come per la precedente) le dimensioni del rettangolo, e avente come valore di ritorno un intero booleano (vero: rettangolo trovato, falso: rettangolo non trovato). La funzione deve poter essere chiamata come segue:

```
if (riconosciRegione(M,nr,nc,r,c,&b,&h)) {
  // stampa messaggio per rettangolo con
  // estremo in (r,c), base b e altezza h
  ...
}
```

Esercizio n.2: **Puntatori e rappresentazione dati**
Competenze: puntatori, codifica dell'informazione, rappresentazioni numeriche
Categoria: il tipo di dato puntatore (Puntatori e strutture dati dinamiche 1.1, 1.2, 1.3)

Si realizzi una funzione che permetta di visualizzare la codifica interna (binaria) di un numero reale, realizzato, in `C`, da un `float`, `double` o `long double`.
Premessa: i tipi `C` `float`, `double` e `long double` (se ne veda, ad esempio, la definizione su https://en.wikipedia.org/wiki/C_data_types) realizzano le specifiche IEEE-754 (https://it.wikipedia.org/wiki/IEEE_754) per i tipi di dato reali in precisione singola, doppia ed estesa/tripla/quadrupla. Si noti che per il tipo `long double` lo standard C non ha una scelta univoca, ma tutti i formati per `long double` hanno 15 bit di esponente.

Il programma `C`:

- usa 3 variabili per numeri reali (`af`, `ad`, `ald`, rispettivamente di tipo `float`, `double` e `long double`)
- determina (utilizzando un numero intero, a scelta del programmatore) se il calcolatore utilizza la codifica little endian o big endian e assegna di conseguenza il valore vero o falso (come intero) a una variabile `bigEndian`
- visualizza (mediante l'operatore `C` `sizeof`) la dimensione (espressa in byte e in bit) delle tre variabili `af`, `ad`, `ald`
- acquisisce da tastiera un numero decimale (con virgola ed eventuale esponente in base 10), assegnandolo alle tre variabili `af`, `ad`, `ald`

- mediante la funzione `stampaCodifica` visualizza la rappresentazione interna del numero nelle tre variabili `af`, `ad`, `ald`.

La funzione `stampaCodifica`, avente prototipo:

```
void stampaCodifica (void *p, int size, int bigEndian);
```

va chiamata tre volte, ricevendo come parametri, rispettivamente il puntatore a una della tre variabili (convertito a `void *`) e la dimensione della variabile:

```
stampaCodifica((void *)&af,sizeof(af),bigEndian);
stampaCodifica((void *)&ad,sizeof(ad),bigEndian);
stampaCodifica((void *)&ald,sizeof(ald),bigEndian);
```

La funzione `stampaCodifica`, utilizzando l'aritmetica dei puntatori, la conoscenza del tipo di codifica e la dimensione ricevuta come parametro, deve stampare il bit di segno, i bit di esponente e i bit di mantissa del numero.

Suggerimenti e/o consigli:
si noti che NON si chiede di rappresentare come numeri esponente e mantissa, ma solo di visualizzarne i bit. Pur essendo possibili varie soluzioni, si consiglia di ricavare la codifica binaria, nella funzione `stampaCodifica`, con la strategia che segue:

- non potendo realizzare in `C` un vettore di bit, si consiglia di leggere/riconoscere il numero ricevuto come vettore di `unsigned char` (ad esempio, un `float` da 32 bit corrisponde a un vettore di 4 `unsigned char`). Si usa l'`unsigned` per limitarsi a numeri senza segno (non negativi). Ogni elemento del vettore va poi decodificato mediante un algoritmo di conversione a binario (riconoscimento dei bit di un numero senza segno). In pratica, il puntatore `p` (di tipo `void *`) andrà internamente assegnato a un puntatore a `unsigned char`
- occorre stampare i bit, separando bit si segno, di esponente e di mantissa, a partire dal più significativo (MSB). A seconda del parametro `bigEndian`, si stabilisce la direzione in cui percorrere i byte del numero. Il parametro `size` serve per sapere dove terminano i bit di esponente ed iniziano quelli della mantissa. Il percorso sui byte può essere realizzato mediante opportuno utilizzo dell'aritmetica dei puntatori, tale da percorrere tutti i byte del dato dal più significativo al meno significativo (o viceversa, a seconda della scelta fatta per la decodifica).

Valutazione: entrambi gli esercizi 1 e 2 saranno oggetto di valutazione
Scadenza: caricamento di quanto valutato: entro le 23:59 del 29/10/2019.