Esercitazione Capitolo 3:

Programmazione orientata agli oggetti

1. Moduli in C

Implementa **un** contatore cui operazioni sono azzera, incrementa e restituisci il valore (come char*). Usa un modulo in C (uso di variabili static nel .c).

2. Tipi opachi in C

Scrivi la definizione di un tipo di dato astratto **Studente** che rappresenta uno studente con un nome (max 30 caratteri), cognome e matricola. Scrivi le seguenti funzioni (segnatura da completare):

- **makeStudent**(char* n , char *c) --> restituisce uno studente con nome n, cognome c e matricola pari a un numero progressivo che dipende dal numero di studenti esistenti

- **printStudente**(studenteRef s) --> stampa cognome nome e matricola

- char* **studentData**(..) → restituisce i dati dello studente come stringa

- **deleteStudente**(studenteRef* s) --> cancella uno studente

Ogni studenti ha anche un elenco di voti (assumi che sia uguale a 20) che viene memorizzato.

- **addVoto**(int x): aggiunge un voto allo studente

- **stampaMedia(…)**: stampa la media dei voti

Definire anche un tipo di dato astratto **Corso** che rappresenta un corso con nome (max 40 caratteri) e un elenco di studenti che frequentano il corso. Scrivi le seguenti funzioni:

- **mkCorso**(char *n) --> restituisce un corso con lista studneti pari a NULL

- **addStudent**(corsoRef c, studenteRef s) --> dato un corso e uno studente, aggiunge lo studente alla lista dei partecipanti al corso (il nuovo studente viene inserito in testa alla lista).

Scrivere un main in cui si creano due studenti e un corso, si inserisce lo studente 1 nel corso. Si mettono dei voti e si cancella lo studente 2.

3. Tipi opachi per array di booleani (bitvector)

Per manipolare dei bitvector in modo sicuro si possono usare i tipi opachi.

Definisci un tipo opaco `bitvector` che ha come campi la dimensione e il contenuto vero e proprio come array di qualche tipo primitivo (ad esempio int[]).

I metodi che la libreria mette a disposizione sono:

- creazione di una bitvector (make) di dimensione n inizialmente tutto falso.
- le operazioni logicche AND, OR, NOT. Le operazioni binarie sono fatte a bit a bit solo se i due bitvector hanno dimensione uguale.
- toString che restituisce una string di 0 e 1 che rappresenta il contenuto del bitvector.
- e un delete che fa da distruttore

Fai un esempio ben commentato in cui utilizzi tutti i metodi sopra.

Ottimizzazione: un char o int in verità può ospitare tanti bit, quindi è inutile usare un intero per ogni bit. Ad esempio se int ha sizeof uguale a 4 byte, sono quindi 32 bit, potrei ospitare 32 bit.