Programmazione e Strutture Dati – Traccia 2
Prima Prova Intercorso 17/04/2018

Da codice etico unisa http://web.unisa.it/uploads/rescue/41/76/codice-etico-e-di-comportamento-unisa.pdf
ART. 43 – VIOLAZIONE DEI DOVERI DEL CODICE - STUDENTI
1. La violazione delle norme del presente Codice da parte degli studenti può dar luogo a sanzioni disciplinari, ai sensi del Regolamento Studenti dell'Ateneo.
2. Quando siano accertate attività tese a modificare indebitamente l'esito delle prove o impedirne una corretta valutazione, il docente o altro preposto al controllo dispone l'annullamento delle prove medesime e la segnalazione al Rettore ai fini dell'attivazione del procedimento disciplinare ai sensi del Regolamento studenti.

Da Regolamento studenti unisa http://web.unisa.it/uploads/rescue/31/19/reg_studenti_2014_web.pdf
ART. 40 – SANZIONI DISCIPLINARI A CARICO DEGLI STUDENTI
1. Le sanzioni che si possono comminare sono le seguenti:
a) ammonizione;
b) interdizione temporanea da uno o più attività formative;
c) esclusione da uno o più esami o altra forma di verifica di profitto per un periodo fino a sei mesi;
d) sospensione temporanea dall'Università con conseguente perdita delle sessioni di esame.
2. La relativa competenza è attribuita al Senato accademico, fatto salvo il diritto dello studente destinatario del provvedimento di essere ascoltato.
3. L'applicazione delle sanzioni disciplinari deve rispondere a criteri di ragionevolezza ed equità, avuto riguardo alla natura della violazione, allo svolgimento dei fatti e alla valutazione degli elementi di prova. Le sanzioni sono comminate in ordine di gradualità secondo la gravità dei fatti.
4. La sanzione è comminata con decreto rettorale.
5. Tutte le sanzioni disciplinari sono registrate nella carriera scolastica dello studente e vengono conseguentemente trascritte nei fogli di congedo.

Esercizio 1 (20 punti). Si implementi, completando il progetto fornito (Esercizio 1) e mediante l'uso di opportune strutture dati, un programma per la gestione di un garage per automobili. Il garage tiene traccia dei seguenti dati:

- Il nome;
- L'indirizzo;
- Il numero di posti disponibili;
- Una lista di automobili caratterizzate da nome (campo unico che include marca e modello), anno di immatricolazione e segmento (compreso tra 1 (mini car) e 6 (luxury car)).

Specificare e realizzare l'ADT Garage che dovrà fornire operatori per:

- Creare il garage;
- Inserire una nuova automobile nella lista;
- Rimuovere un'automobile dalla lista, data la sua posizione;
- Spostare in avanti (*forward*) un'automobile che nella lista occupa la posizione *i*, scambiandola con l'automobile in posizione *i+1*;
- Estrarre dalla lista delle auto (clonandola) la sottolista composta dalle auto di un dato segmento;
- Visualizzare i dati del garage, con la lista delle auto.

Si producano i seguenti deliverable (i punti associati a ciascun deliverable sono indicativi):

1. Su carta: specifica dell'ADT (Garage e Car) – 4 punti;
2. Sulla chiavetta USB, i seguenti moduli software:
    a. Garage – 4 punti;
    b. Car – 2 punti;
    c. Lista linkata (completare lo stub esistente, con l'aggiunta degli operatori generici richiesti dal programma) – 5 punti;
    d. Programma principale con menu testuale per effettuare le operazioni richieste – 4 punti;
    e. Makefile (completare lo stub esistente) per compilare e linkare il programma – 1 punto.

Esercizio 2 (15 punti). Si implementi e si testi, completando il progetto fornito (Esercizio 2) e mediante l'uso della struttura dati vettore, un programma che, letto da input un vettore di stringhe, ricopi in un nuovo vettore gli elementi pari (il secondo, il quarto, …) del vettore il cui valore è minore (secondo l'ordine lessicografico) rispetto all'elemento che lo precede nel vettore originario. Per esempio, dato il seguente input:

```
aa bb dd cc ee ff
```

Produca il seguente output:

```
cc
```

Si producano i seguenti moduli software (i punti associati a ciascun deliverable sono indicativi):

a. Item-String (completare lo stub esistente, con l'aggiunta delle funzioni il cui prototipo è specificato nell'header file) – 4 punti;
b. Vettore (completare lo stub esistente, con l'aggiunta delle funzioni il cui prototipo è specificato nell'header file) – 4 punti;
c. Driver per il testing della funzione `cloneEvenItems` – 4 punti;
d. File di input ed oracolo per il testing della funzione `cloneEvenItems` - 2 punti;
e. Makefile (completare lo stub esistente) per generare il programma e il driver per il testing – 1 punto.