Compito di Programmazione
29 Maggio 2023

Nota Bene. Ogni esercizio deve essere svolto su un foglio diverso.
Scrivere Nome, Cognome e Matricola su ogni foglio.

1. **(punti 8)** Si considerino le seguenti strutture dati

```
struct Elem {
    int val ;
    int next ;
}

struct List {
    int init ;
    Elem vect[l] ;
}
```

dove `l` è una costante. Una variabile `A` di tipo `List` rappresenta una *lista* in cui l'indice del primo elemento è memorizzato in `A.init` ed il primo elemento si trova in `A.vect[A.init]` mentre l'elemento successivo si trova ad indice `(A.vect[A.init]).next` (e così via). Se `A.init == -1`, la lista è vuota; l'ultimo elemento della lista ha campo `next` uguale a `-1`. Una lista `A` può essere *circolare*, nel senso che, scorrendo gli elementi della lista, non si trova mai un elemento il cui campo `next` è uguale a `-1`.

   - definire una funzione *ricorsiva* `last_index` che ritorna l'indice dell'ultimo elemento di una lista `L`; ritorna `-1` se non esiste l'ultimo elemento (lista vuota o lista circolare);
   - definire una funzione `get_min` che prende una lista *non vuota* e ritorna il minimo valore che è memorizzato nella lista. Attenzione: la lista può essere circolare.

2. **(punti 8)** Si vuole rappresentare un magazzino contenente diversi tipi di prodotti. Ogni prodotto é caratterizzato dal nome (array di char), dalla quantitá disponibile (intero) e dal prezzo unitario (double). Definire le strutture dati necessarie per rappresentare il magazzino (come lista di prodotti) e le seguenti funzioni:

   - `rimuovi_quantita` che prende come parametri il magazzino e il nome del prodotto, e rimuove un'unitá del prodotto dal magazzino. Se, dopo aver rimosso il prodotto, la quantitá é pari a 0 bisogna rimuovere il prodotto dal magazzino. Questa funzione deve restituire la lista aggiornata.
   - `valore_magazzino` che prende come parametro il magazzino e calcola il valore totale del magazzino come la somma dei prodotti della quantitá disponibile per il prezzo unitario di ciascun prodotto presente nel magazzino.
   - `nome_prodotto` che prende come parametro il magazzino e restituisce il nome del prodotto con la quantitá maggiore.

   [**N.B.** Si possono usare funzioni ausiliarie se definite.]

3. **(punti 8)** Un parcheggio è caratterizzato da un insieme di posti auto e da un costo orario. Un posto auto è a sua volta caratterizzato dalla targa della macchina che lo occupa, l'ora di inizio del parcheggio e da un flag che indica se è libero o meno. Si rappresenti un posto auto tramite `struct` e le ore come semplici interi, da 0 a 23. Per semplicità si gestisca il caso di parcheggi in giornata, non giorni multipli o consecutivi. Si implementi la classe `Parcheggio`, il relativo costruttore e i seguenti metodi:

   - `occupa_posto()`: il quale riceve come parametri la targa e l'ora di inizio del parcheggio, cerca un posto libero e se lo trova lo occupa e salva targa e orario.
   - `libera_posto()`: il quale riceve come parametri la targa e l'ora di fine parcheggio, cerca l'ora di inizio del parcheggio per quella targa, se la trova, libera il posto, calcola il costo del parcheggio e lo restituisce.

   Esistono anche i parcheggi che hanno delle fasce di orario gratuite, i quali sono caratterizzati da un'ora dalla quale si inizia a pagare (es. 8:00) e un'ora oltre la quale non si paga (es. 20:00). Si implementi la sottoclasse `ParcheggioConOrari` e si modifichino i metodi sfruttando l'ereditarietà in modo tale che, l'ora di inizio non sia inferiore all'ora in cui si inizia a pagare in quel parcheggio e l'ora di fine non vada oltre l'ora in cui si smette di pagare in quel parcheggio.
   In tutto l'esercizio 3 non è consentito usare le liste.