//

// APPELLO del 14/1/2022: Testo e possibili soluzioni

//

/* ESERCIZIO 1

1.A (2 PUNTI) Realizzare due struct indirizzo e cliente: Indirizzo contiene: via, numero civico, CAP, città Cliente contiene: codice fiscale, cognome, nome, indirizzo

*

1.B (2 PUNTI) Realizzare una funzione che verifichi se due clienti abitano nella stessa zona della città (da verificare tramite il CAP)*/

/* ESERCIZIO 2

Consideriamo il tipo di dato “coda di Elem” e un’implementazione basata su vector

2.A (2 PUNTI) Produrre i prototipi (o interfacce) delle 3 funzioni principali

- enqueue (inserisci elemento in fondo alla coda)

- dequeue (elimina elemento dalla testa della coda)

- front (accedi in lettura e restituisci il prossimo elemento nella coda)

2.B (2 PUNTI) Implementare la funzione dequeue NB le code funzionano secondo il meccanismo FIFO (first in first out): esce dalla coda l’elemento che e’ stato in coda piu’ a lungo Una VERSIONE 1 qui di seguito permette di rispondere alla domanda in modo molto semplice (la difficoltà si sposterebbe sulla funzione enqueue, con inserimento di un elemento dalla testa)

/* ESERCIZIO 3

Considerate le liste collegate semplici: typedef struct cell {

int head;

cell *next;

} *lista;

3.A (2.5 PUNTI) Realizzare una funzione ricorsiva che permetta di contare il numero di elementi di una lista

/* 3.B (2.5 PUNTI) Realizzare una funzione booleana che restituisce true se tutti gli elementi della lista sono pari, false altrimenti.

Trattare in modo opportuno il caso lista vuota

/* VERSIONE RICORSIVA */

/* VERSIONE ITERATIVA*/