Risposte Esame di Programmazione - 2021-12-22

Esercizio 1A

Realizzare due struct indirizzo e cliente:

```
struct Indirizzo {
    std::string via;
    int numero_civico;
    std::string CAP;
    std::string citta;
};

struct Cliente {
    std::string codice_fiscale;
    std::string cognome;
    std::string nome;
    Indirizzo indirizzo;
};
```

Esercizio 1B

Realizzare una funzione che verifichi se due clienti abitano nella stessa zona della citta (da verificare tramite il CAP).

```
bool stessaZona(Cliente c1, Cliente c2) {
    return c1.indirizzo.CAP == c2.indirizzo.CAP;
}
```

Risposte Esame di Programmazione - 2021-12-22

Esercizio 2A

Consideriamo il tipo di dato ?coda di Elem? e un?implementazione basata su vector. Produrre i prototipi (o interfacce) delle 3 funzioni principali:

```
void enqueue(std::vector<Elem> &coda, Elem elemento);
void dequeue(std::vector<Elem> &coda);
Elem front(const std::vector<Elem> &coda);
```

Esercizio 2B

Implementare la funzione dequeue:

```
void dequeue(std::vector<Elem> &coda) {
    if (!coda.empty()) {
        coda.erase(coda.begin());
    }
}
```

Esercizio 3A

Realizzare una funzione ricorsiva che permetta di contare il numero di elementi di una lista.

```
int contaElementi(lista l) {
    if (l == NULL) {
        return 0;
    } else {
```

```
        return 1 + contaElementi(l->next);
    }
}
```

Esercizio 3B

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.

```
bool tuttiPari(lista l) {
    if (l == NULL) {
        return true;
    } else {
        return (l->head % 2 == 0) && tuttiPari(l->next);
    }
}
```