Risposte Esame di Programmazione - 2021-12-22

Esercizio 1A

Realizzare la struct casella. Ogni casella è descritta dalla posizione (riga, colonna), e dallo stato (per es "libero", "muro", "occupato").

```
struct Casella {
    int riga, colonna;
    std::string stato;
};
```

Esercizio 1B

Realizzare la funzione booleana bool Adiacente (casella c1, casella c2) che restituisce true se c1 e c2 sono vicini (sinistra-destra o sopra-sotto).

```
bool Adiacente(Casella c1, Casella c2) {
    return (abs(c1.riga - c2.riga) == 1 && c1.colonna == c2.colonna) ||
           (abs(c1.colonna - c2.colonna) == 1 && c1.riga == c2.riga);
}
```

Esercizio 2A

Produrre una funzione di inserimento di una casella nel percorso, dopo aver verificato che sia interna al tabellone e che sia adiacente alla casella precedente.

```
void InserisciCasella(std::vector<Casella> &percorso, Casella c) {
    const int DIM = 10; // Supponiamo una dimensione del tabellone
```

```
    if (c.riga < 0 || c.riga >= DIM || c.colonna < 0 || c.colonna >= DIM) {
        throw std::out_of_range("Casella fuori dal tabellone");
    }
    if (percorso.empty() || Adiacente(percorso.back(), c)) {
        percorso.push_back(c);
    } else {
        throw std::invalid_argument("Casella non adiacente alla precedente");
    }
}
```

Esercizio 2B

Realizzare una funzione che calcoli l?ampiezza massima del percorso (ossia la distanza dalla casella più a sinistra a quella più a destra).

```
int AmpiezzaMassima(const std::vector<Casella> &percorso) {
    if (percorso.empty()) return 0;
    int min_col = percorso[0].colonna;
    int max_col = percorso[0].colonna;
    for (const auto &c : percorso) {
        if (c.colonna < min_col) min_col = c.colonna;
        if (c.colonna > max_col) max_col = c.colonna;
    }
    return max_col - min_col + 1;
}
```

Risposte Esame di Programmazione - 2021-12-22

Esercizio 3A

Definire il tipo di dato insieme basato su liste semplici (tipo base: interi).

```
struct Nodo {
    int valore;
    Nodo* next;
};

typedef Nodo* Insieme;
```

Esercizio 3B

Realizzare la funzione che effettui l?inserimento di un elemento nuovo, nel rispetto delle proprietà degli insiemi.

```
void inserisciElemento(Insieme &insieme, int elemento) {
    Nodo* nuovo = new Nodo{elemento, nullptr};
    if (!insieme) {
        insieme = nuovo;
    } else {
        Nodo* current = insieme;
        while (current->next && current->valore != elemento) {
            current = current->next;
        }
        if (current->valore != elemento) {
```

```
            current->next = nuovo;
        }
    }
}
```

Esercizio 3C

Realizzare una funzione ricorsiva che stampi gli elementi dell?insieme.

```
void stampaInsieme(const Insieme &insieme) {
    if (!insieme) return;
    std::cout << insieme->valore << " ";
    stampaInsieme(insieme->next);
}
```