

Esercizio 1

Data la seguente porzione di programma rispondere alle domande corrispondenti:

```
void f(int& a) {if (a%2==0) a++; else a-=2;}
int &g(int& a) {int b=a; return b;}

int main() {

int* matricola = new int[6] {...inserisci la tua matricola...};

// 1. Le istruzioni seguenti sono corrette? Se si, cosa viene stampato?
f(g(matricola[2]));
cout << matricola[2] << endl;

// 2. Le istruzioni seguenti sono corrette? Se si, cosa viene stampato?

int* p = matricola + 4;
int* q = matricola + 3;
p = q;
q = p;
*p=0;
*q=9;
cout << *(matricola+3) << " " << *(matricola+4) << endl;

// 3. La seguente istruzione è corretta? Se si, cosa viene stampato?

cout << matricola [matricola +2] << endl;

// 4. Selezionare TUTTE le operazioni necessarie per deallocare la memoria allocata nel
programma
// a. delete[] matricola;
// b. delete p;
// c. for (int i=0; i < 6; i++) {delete matricola[i];}
// d. delete matricola;
// e. delete q;

return 0;
}
```

Esercizio 2

Modellare e implementare una classe `Webpage` che modelli una pagina web. Ciò che ci interessa memorizzare è il titolo della pagina, il testo contenuto in essa e l'insieme dei suoi link (collegamenti ad altre `Webpage`). Si richiede inoltre di poter leggere queste informazioni dall'esterno della classe, da cui si deve anche poter aggiungere nuovi link ma senza la possibilità di rimuoverli, mentre il testo e il titolo della pagina, una volta creata, devono essere immutabili. Infine, si richiede di definire un operatore di confronto tra due `Webpage`, ritenute uguali se hanno lo stesso titolo.

Si richiede poi di implementare una classe `Website` che modelli la struttura di un sito web. Un `Website` è sostanzialmente un insieme di `Webpage` collegate tra loro e da cui si accede tramite una `Webpage` specifica detta `homepage`.

All'interno di Website si richiede di implementare un metodo

```
int find_webpage(const Webpage& target) const { ... }
```

il quale, presa in input una Webpage, restituisce il minimo numero di link da attraversare per trovare la pagina target e -1 se target non appartiene al sito.

Esercizio 3

Scrivere una funzione `f` che, preso in input un Grafo orientato `g`, restituisca `true` se e solo se risultano verificate tutte le seguenti proprietà:

- Il grado uscente di ogni nodo di `g` è minore del numero di nodi di `g` che hanno al più un arco entrante
- Il numero totale di archi di `g` è pari
- `g` non ha autoarchi (archi che vanno da un nodo a sé stesso)

La classe `Grafo` dispone dei seguenti metodi pubblici (dove `g` è un'istanza della classe `Grafo`):

- `g.n()` che restituisce il numero di nodi del Grafo
- `g.m()` che restituisce il numero di archi del Grafo
- `g(i, j)` che restituisce `true` se esiste un arco dal nodo `i` al nodo `j` e `false` altrimenti

Nota: Verrà assegnato un piccolo bonus a chi svolgerà l'esercizio in modo da 'evitare', quando possibile, di verificare le condizioni computazionalmente più onerose (la complessità dei 3 metodi pubblici della classe `Grafo` è $O(1)$).

Esercizio 4

Su una scacchiera 8×8 , il cavallo è un pezzo che si muove ad "L", spostandosi sempre di una cella in orizzontale e di 2 in verticale o viceversa, in qualsiasi direzione. I movimenti del cavallo possono essere codificati dalle seguenti coppie di interi: $(-2, -1)$, $(-2, +1)$, $(-1, -2)$, $(-1, +2)$, $(+1, -2)$, $(+1, +2)$, $(+2, -1)$, $(+2, +1)$

Ad esempio, se siamo nella cella di riga 4 e colonna 3, seguendo il primo movimento della lista finiremo sulla cella $(2, 2)$, mentre seguendo il secondo movimento arriveremo in $(2, 4)$ e così via.

Scrivere un programma che prenda in input:

1. la posizione iniziale di un cavallo su scacchiera 8×8 (`pair<unsigned, unsigned>`)
2. un insieme di caselle proibite (`vector<pair<unsigned, unsigned> >`)
3. una casella di destinazione (`pair<unsigned, unsigned>`)

Il programma deve determinare se è possibile far arrivare il cavallo dalla sua posizione iniziale alla destinazione, considerando che esso non può fermarsi sulle caselle proibite (ma può attraversarle all'interno di una singola mossa). Naturalmente, non è ammesso che il cavallo si muova al di fuori della scacchiera. Inoltre, non è possibile che il percorso del cavallo preveda di passare sulla stessa casella più di una volta. Se esiste una soluzione dovrà stampare in output l'insieme delle caselle da attraversare per arrivare a destinazione, altrimenti dovrà stampare "IMPOSSIBILE".