Fondamenti di Programmazione 2
(Corso di Laurea in Informatica)
Esame del 18/02/2025

UNIVERSITÀ DELLA CALABRIA

Esercizio 1

Data la seguente porzione di programma rispondere alle domande corrispondenti:

```
int main() {
    int* matricola = new int[6]  {//Inserire qui la tua matricola };

    // 1. La seguente istruzione è corretta? Motivare la risposta
    if (matricola.size() > 4) {
        cout << "OK" << endl;
    }
    // Fine blocco codice relativo a domanda 1

    // 2. La seguente istruzione è corretta? Se si, cosa stampa?
    int * p = new int[3];
    for (int i = 0; i < 3; i++) {
        p[i] = matricola[i*2];
        cout << p[i];
    }
    cout << endl;
    // Fine blocco codice relativo a domanda 2

    // 3. Perché la seguente coppia di istruzioni non è corretta?
    *(matricola + 2) = 6;
    matricola[matricola[2]] = 1;
    // Fine blocco di codice relativo a domanda 3

    // 4. Selezionare TUTTE le istruzioni necessarie per deallocare la memoria allocata
    // nel programma
    //       a. delete matricola;
    //       b. delete p;
    //       c. delete[] matricola;
    //       d. delete[] p;
    //       e. for (int i=0; i < 6; i++) {delete matricola[i];}
    //       f. for (int i=0; i < 3; i++) {delete p[i];}

    return 0;

}
```

Esercizio 2

Un salone di parrucchieri, in cui lavorano un numero imprecisato di dipendenti, ha bisogno di un software per la gestione delle prenotazioni. A tal fine, si richiede di modellare ed implementare una classe `CodaDalParrucchiere` che tenga traccia delle prenotazioni assegnate a ciascun parrucchiere (rappresentato come stringa) ed esponga le seguenti funzionalità:

- Un costruttore che prenda in input i parrucchieri che lavorano nel salone (tramite `vector<string>`)
- Aggiungere una prenotazione: il metodo riceve in input un cliente (anch'esso rappresentato come stringa) e un parrucchiere. Quando un cliente vuole prenotare una prestazione, il programma deve cercare di accontentarlo assegnandogli il parrucchiere da lui scelto. Tuttavia, nel caso in cui il parrucchiere desiderato abbia già più di `10` clienti in coda, la prenotazione sarà assegnata al parrucchiere con meno clienti in coda. Se non ci sono parrucchieri disponibili ad accoglierla, il sistema deve restituire `false` (`true` altrimenti).
- Cancellare una prenotazione: il metodo riceve il nome di un cliente `c` e rimuove la sua prenotazione, se e solo se essa non è imminente (consideriamo imminente la prenotazione di `c` se prima di `c` ci sono meno di `3` clienti in fila). Il metodo restituisce `true` se la prenotazione è stata correttamente cancellata e `false` altrimenti.

- Stimare il tempo d'attesa: che restituisca, in minuti, il tempo che dobbiamo attendere prima di sederci sulla poltrona, considerando che ogni parrucchiere impiega mediamente `15` minuti per ogni cliente.

Vogliamo poi implementare una seconda classe, `CodaDalParrucchiereAccurata`, in cui i meccanismi di prenotazione restano invariati ma la stima dell'attesa prende in considerazione anche la velocità media di lavoro del parrucchiere (minuti impiegati per un cliente). A tal fine, il costruttore di questa nuova classe riceve in input una `map<string,unsigned>`, dove ogni chiave rappresenta un parrucchiere e il valore corrispondente indica il numero di minuti con cui il parrucchiere, in media, processa un singolo cliente. In questa nuova classe terremo in considerazione questo nuovo dato per stimare i tempi d'attesa con maggiore accuratezza.

Si noti che la scelta delle strutture dati usate per risolvere l'esercizio ne influenza (positivamente o negativamente) la valutazione.

Esercizio 3

Dato un Grafo orientato `g`, un nodo di partenza `s`, un nodo di destinazione `t` e un arco `(i,j)` del grafo, scrivere una funzione

```
bool f(const Grafo&g, unsigned s, unsigned t, pair<unsigned,unsigned> edge) {
      …
}
```

che restituisca `true` se e solo se, dopo aver invertito l'orientamento dell'arco `(i,j)`, il cammino minimo da `s` a `t` diventa più breve.

Esercizio 4

Una matrice `n*n`, rappresentata tramite `vector<vector<int>> pf`, può contenere al suo interno un'istanza del gioco **Minesweeper** (prato fiorito), tale che:

- Se `pf[i][j] = -1`, nella cella `(i,j)` troveremo una bomba
- Se `pf[i][j] = x`, con `0 <= x <= 8`, avremo che `x` celle adiacenti a `(i,j)` contengono una bomba
- Se `pf[i][j] = 9`, avremo che la cella `(i,j)` è indeterminata (potrebbe o non potrebbe contenere una bomba)

Nel caso in cui tutte le celle di `pf` siano diverse da 9, diciamo che la matrice è **totalmente valorizzata**, e, al contrario, diciamo che la matrice è **parzialmente valorizzata.**

Si richiede di scrivere un programma che, presa in input un'istanza di **Minesweeper** parzialmente valorizzata e un numero intero positivo `k`, restituisca `true` se e solo se è possibile creare un'istanza totalmente valorizzata di **Minesweeper** tale che:

- Contenga esattamente `k` bombe.
- Per ogni cella non contenente bombe, il numero contenuto in essa rispecchi l'effettivo conteggio delle sue bombe adiacenti.
- Le celle che in input non sono indeterminate devono contenere il valore di partenza.

In caso contrario, il programma deve restituire `false`.