Fondamenti di Programmazione 2
(Corso di Laurea in Informatica)
Esame del 4/07/2024

UNIVERSITÀ DELLA CALABRIA

Esercizio 1

Data la seguente porzione di programma rispondere alle domande corrispondenti:

```
int main() {
//scrivi sul foglio la tua matricola
int* matricola = new int[6]{..la tua matricola..};

int* nuova = new int[6];
for (int i=0; i < 6; i++) {
    nuova[i] = (matricola[i]+1)%10;
}

nuova = matricola;

// 1. La seguente istruzione è corretta? Se no, perché?
*matricola.push_back(7);

// 2. La seguente istruzione è corretta? Se si, cosa stampa?
cout << *(nuova + 3) << endl;

// 3. La seguente istruzione è corretta? Se si, cosa stampa?
cout << *(matricola + 3) << endl;

// 4. A questo punto, è possibile liberare tutta la memoria allocata?
// A. Si, è sufficiente l'istruzione delete[] nuova;
// B. Si, ma sono necessarie entrambe delete[] nuova; e delete[] matricola;
// C. No, l'array puntato originariamente da nuova non è più raggiungibile.
// D. Si, nuova è stato cancellato, per cui è sufficiente delete[] matricola;
}
```

Esercizio 2

Implementare una classe **ArrayWrapper** che semplifichi l'utilizzo degli array dinamici in C++. In particolare, è necessario implementare la seguente interfaccia pubblica:

```
template<class T>
class ArrayWrapper {
public:
  ArrayWrapper();
  ArrayWrapper(const ArrayWrapper<T>& _w);
  ~ArrayWrapper();
  ArrayWrapper<T>& operator=(const ArrayWrapper& w);
  T back() const;
  void push_back(const T& _value);
  void pop_back();
  unsigned size();
  T& operator[](int i);
  T operator[](int i) const;
}
```

Ai fini dell'esercizio, non è consentito far uso di classi della libreria standard.
Implementare almeno tutti i metodi in grassetto. L'implementazione corretta degli altri metodi potrà dare diritto ad un bonus di massimo 2 punti a patto che i metodi obbligatori siano corretti.
Risulta necessario infine implementare una strategia per evitare l'accesso ad indici non validi.
Si possono creare tutti i metodi "di supporto" e le componenti dati che si ritengono necessari, ma essi non possono essere esposti al di fuori della classe.

UNIVERSITÀ DELLA CALABRIA

Esercizio 3

Scrivere una funzione che preso in input un `vector<string>` e un numero intero positivo **k** stampi in ordine decrescente di occorrenza le prime **k** parole più frequenti tra quelle in input.

La funzione **dovrà** necessariamente fare uso della struttura dati `std::map` e della funzione `std::sort`. In modo corretto, naturalmente.

Esercizio 4

Sei stato incaricato di scegliere i pezzi per il DJ set di una celebre festa universitaria. Dovrai rispettare i seguenti requisiti:

- La durata totale dovrà essere di esattamente **k** minuti
- Includere almeno una traccia per ciascun genere tra tutti quelli presenti nelle tracce disponibili

Scrivere una funzione che presi in input un `vector<Traccia>` e un intero **k** restituisca un `vector<Traccia>` con i pezzi selezionati.

Si può assumere che la classe `Traccia` sia già implementata e esponga la seguente interfaccia pubblica:

```
struct Traccia {
     const string titolo;
     const unsigned m;
     const int g;
};
```

in particolare, **m** indica la durata in minuti di una traccia; l'intero **g** codifica un genere musicale. Non si può assumere che l'insieme dei generi sia noto in anticipo.