Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 15/02/2024

Nome..... Cognome..... Matricola.....

Esercizio Funzione.

Definire un template di funzione

`template <class T> list<const iostream*> compare(vector<ostream*>&, vector<const T*>&)`

con il seguente comportamento: in ogni invocazione `compare(v,w)`,

1. se `v` e `w` non contengono lo stesso numero di elementi allora viene sollevata una eccezione di tipo `string` che rappresenta la stringa vuota;
2. se `v` e `w` contengono lo stesso numero di elementi allora per ogni posizione `i` dentro i bounds dei due vettori `v` e `w`:
   (a) se `*v[i]` è un `fstream` ed è dello stesso tipo di `*w[i]` allora: (i) il puntatore `v[i]` viene inserito nella lista che la funzione deve ritornare; (ii) i puntatori `v[i]` e `w[i]` vengono rimossi dai vettori che li contengono;
   (b) se `*w[i]` è uno `stringstream` in stato good e `*v[i]` e `*w[i]` sono di tipo diverso allora il puntatore `w[i]` viene inserito nella lista che la funzione deve ritornare.

```
template <class T> list<const iostream*> compare(vector<ostream*>& v, vector<const T*>& w){
    list<const iostream*> lista;
    for(int i = 0; i < v.size(); i++){
        if(v.size() ≠ w.size()){
            throw std::string("");
        }
        else{
            if(dynamic_cast<fstream*>(const_cast<iostream*>(*v[i]) &&
            typeid(*v[i]) == typeid(*w[i])){
                lista.push_back(dynamic_cast<iostream*>(v[i]));

                ostream* temp = *v[i]; // not const, safe
                T* t = const_cast<T*>(*w[i]); // const, keep in mind

                // erase = cancellazione nel vector
                v.erase(v.begin() + i); // di default, va alla
                // posizione subito dopo
                v.erase(t);

                delete temp;    // deallochiamo le variabili dinamiche
                delete t;
            }

            stringstream *s = dynamic_cast<fstream*>
            (const_cast<stringstream*>(*v[i]));

            if(s && s→good() && typeid(*v[i]) ≠ typeid(*w[i])){
                lista.push_back(dynamic_cast<iostream*>(v[i]));
            }
        }
    }
    return lista;
}
```