Esercizio Funzione Definire un template di funzione `template<class T> list<const iostream*> fun(vector<ostream*>&)` con il seguente comportamento: in ogni invocazione `fun(v)`, per ogni puntatore `p` elemento (di tipo `ostream*`) del vector `v`:

1. se `p` non è nullo e `*p` è un `fstream` che non è nello stato good (ovvero stato 0, con tutti i bit di errore spenti), allora `p` diventa nullo;
2. se `p` non è nullo e `*p` è uno `stringstream` nello stato good, allora `p` viene inserito nella lista che la funzione deve ritornare;
3. se la lista che la funzione deve ritornare è vuota allora la funzione solleva una eccezione di tipo `T`, altrimenti ritorna la lista.

```
template<class T>std::list<const std::iostream*> fun(std::vector<std::ostream*>& v){
        std::list<const std::iostream*> lista;
        // Variante scorrimento classico
        for(auto p = v.begin(); p != v.end(); ++p){
                if(p && dynamic_cast<fstream*>(*p) && !dynamic_cast<fstream*>(*p)->good()){
                        p = nullptr;
                        // Cancellazione di "p" -> const iostream*
                        iostream* i = const_cast<iostream*>(*p);
                        p = v.erase(i);
                        delete i;
                }
                if(p && dynamic_cast<stringstream*>(*p) &&
                !dynamic_cast<stringstream*>(*p)->good()){
                        lista.push_back(dynamic_cast<const iostream*>(*p));
                }
                // se lista vuota
                if(lista.empty())
                        throw T();
                return lista;
        }
        // std::list<const std::iostream*>::const_iterator cit

        // Variante con "i"
        for(int i = 0; i < v.size(); i++){
                if(v[i] && dynamic_cast<fstream*>(*v[i]) && dynamic_cast<fstream*>(*v[i]))
                ->good()){
                        v[i] = nullptr;
                }
        }
}
```