

Si considerino le seguenti dichiarazioni di classi di qualche libreria grafica, dove gli oggetti delle classi `Container`, `Component`, `Button` e `MenuItem` sono chiamati, rispettivamente, contenitori, componenti, pulsanti ed entrate di menu.

```
class Component;

class Container {
public:
    virtual ~Container();
    vector<Component*> getComponents() const;
};

class Component: public Container {};

class Button: public Component {
public:
    vector<Container*> getContainers() const;
};
```

```
class MenuItem: public Button {
public:
    void setEnabled(bool b = true);
};

class NoButton {};
```

Assumiamo i seguenti fatti.

1. Il comportamento del metodo `getComponents()` della classe `Container` è il seguente: `c.getComponents()` ritorna un vector di puntatori a tutte le componenti inserite nel contenitore `c`; se `c` non ha alcuna componente allora ritorna un vector vuoto.
2. Il comportamento del metodo `getContainers()` della classe `Button` è il seguente: `b.getContainers()` ritorna un vector di puntatori a tutti i contenitori che contengono il pulsante `b`; se `b` non appartiene ad alcun contenitore allora ritorna un vector vuoto.
3. Il comportamento del metodo `setEnabled()` della classe `MenuItem` è il seguente: `mi.setEnabled(b)` abilita (con `b==true`) o disabilita (con `b==false`) l'entrata di menu `mi`.

Definire una funzione `Button** Fun(const Container&)` con il seguente comportamento: in ogni invocazione `Fun(c)`

1. Se `c` contiene almeno una componente `Button` allora
ritorna un puntatore alla prima cella di un array dinamico di puntatori a pulsanti contenente tutti e soli i puntatori ai pulsanti che sono componenti del contenitore `c` ed in cui tutte le componenti che sono una entrata di menu e sono contenute in almeno 2 contenitori vengono disabilite.
2. Se invece `c` non contiene nessuna componente `Button` allora solleva una eccezione di tipo `NoButton`.