Esercizi per il C++ - 2 parte

1. slicing

Dichiara una classe base e una sua derivata. Tipo:

```
class A {
   int foo;
};
```

```
class B : public A
{
   int bar;
};
```

Cosa succede se fai (in un main):

```
B b;
A a = b;
```

Aggiungi un metodo M ad A e prova a ridefinirlo in B. Cosa succede se chiami il metodo M? Quale viene eseguito? Prova a metterelo virtual, cambia qualcosa?

La stessa cosa con i metodi. Cosa succede sa fai:

```
void wantAnA(A myA){
   myA.M()
}

B derived;
wantAnA(derived);
```

Quale viene eseguito? Come dovresti cambiare il metodo `wantAnA` per attivare il polimorfismo?

2. ereditarietà multipla

Fai un esempio di eredità multipla.

Ridefinisci qualche metodo nella classe derviata e chiama i metodi nelle classi base in modo che ci sia name clash. Come lo risolvi?

Aggiungi anche una classe in modo di avere una configurazione a damiante (senza derivazione virtual). Fai un modo di avere qualche problema di suplicazione dei campi base. Come la risolvi?

5. STL

STL1: declare a vector of integer values, stores five arbitrary values in the vector and then print the single vector elements to cout.

STL2: declare a vector of string values, asks to the user to insert a sentence of one or more words, store each word in the vector and then print the sentence in reverse order

STL3: come STL2 ma con un iteratore al contrario.

STL4: scrivi un metodo che legge parole da un file, le memorizza in una coda e le ristampa.

STL5: scrivi una piccola applicazione che gestisce una rubrica telefonica in cui usi:

- una map: array associativo per memorizzare nomi e numeri di telefono

- la funzione find (che prende due iteratori) per cercare un certo numero o persona

- la funzione copy per copiare l'intera rubrica a video con una sola istruzione (senza ciclo for)

copy(myVector.begin(), myVector.end(), ostream_iterator<int>(cout, " "));