Define an abstract class Entry with an unique sequential number, a description, proper setters and getters for the attributes, the virtual destructor and the pure virtual method getContents returning a string with all the entry contents.
Define the following classes inherited from the Entry:

- PhoneEntry extending the Entry with a phone number,
- EmailEntry extending the Entry with an e-mail address,
- AddressEntry extending the Entry with an address containing a city, a street and a house number.

Override the virtual method for the classes returning all the features of the particular entry together with the description and the sequential number. Implement the necessary constructors, destructors, getters, setters, other methods and exceptions.

Define the class ContactBook with a dynamic list of the entries labeled with some short names. The ContactBook should be ordered by the entry names. Implement public methods of the class enabling to add a new entry of an arbitrary type to the book and to remove all the entries from the book. Overload the indexing operator ([]) for the book to have a direct access to the particular entry in the book by the entry name (throwing an exception if it doesn't exist). Overload the shift-left operator (<<) printing the contents of all the entries in the book. Add other members which are necessary to implement the class.

Write a program which tests all the class capabilities, in particular the following code should be enabled:

```
ContactBook contacts("My contacts");

contacts.add("Johny", new PhoneEntry("John Smith", 100200300));
contacts.add("Lisa", new EmailEntry("Lisa Wood", "lisa@gmail.com"));
contacts.add("Andy", new AddressEntry("Andrew Fox", "Warsaw", "Green St.", 7));

cout << contacts;
//result (ordered):
//Andrew Fox: Warsaw, Green St. 7 (pos. 3)
//John Smith: phone 100200300 (pos. 1)
//Lisa Wood: e-mail lisa@gmail.com (pos. 2)

try
{

  cout << contacts["Andy"].getContents() << endl;
  //result:
  //Andrew Fox: Warsaw, Green St. 7 (pos. 3)
  cout << contacts["Name"].getContents() << endl;
  //NameError exception

}
catch(ContactBook::NameError &e)
{

  cout << e.what(); //Entry named "Name" not found.

}

contacts.clear();
```