Part 1:

1. **Base Class: ListItem**
   Create an abstract class "`ListItem`" that represents a generic list item. This class must contain a pure virtual method "`toString()`" that returns a string representation of the object.
2. **Derived Class: StringItem**
   Create the derived class `StringItem` that inherits from the `ListItem` class. This class represents an item in the form of a string. Be sure to add an explicit default constructor. Add the `toString()` method, which returns a string representation of a StringItem.
3. **Derived Class: NumberItem**
   Create the derived class `NumberItem` that inherits from the `ListItem` class. This class represents an item in the form of a number (double). Be sure to add an explicit default constructor. Add the `toString()` method, which returns a string representation of a `NumberItem`.
4. Implement the overloading of the << operator for an object of the `ListItem` class.

Main Part 1:

If all the classes are properly defined, the code of the main function should display the following output:

Main	int main() {
	// part 1
	ListItem* pstr = new StringItem("Hello");
	cout << " Displaying a string element:" << *pstr << endl;
	ListItem* pnbr = new NumberItem(5);
	cout << " Displaying a number element:" << *pnbr << endl;
	delete pstr;
	delete pnbr;
	return 0;
	}

Display	Displaying a string element: Hello Display a number element: 5

Part 2:

Implement the necessary methods to enable deep copying of a `ListItem` object.

Main Part 2:

The code of the main function must display the following output:

Main	int main() {
	// part 1
	ListItem* pstr = new StringItem("Hello");
	cout << " Displaying a string element:" << *pstr << endl;
	ListItem* pnbr = new NumberItem(5);
	cout << " Displaying a number element:" << *pnbr << endl;
	delete pstr;
	delete pnbr;
	// part 2
	ListItem* pstr_copy = pstr->copy();
	cout << " Displaying a string element after copy: " << *pstr_copy << endl;
	ListItem* pnbr_copy = pnbr->copy();
	cout << " Displaying a number element after copy " << *pnbr_copy<< endl;
	delete pstr;
	delete pnbr;
	delete pstr_copy;
	delete pnbr_copy;
	return 0;
	}

Display	Displaying a string element: Hello Display a number element: 5

Part 3:

Create the class `PythonList` that encapsulates a vector of pointers to `ListItem` objects. Implement in this class a method `addItem` that allows adding a pointer to a `ListItem` object to the list, as well as the following methods:

- ✓ A copy constructor.
- ✓ A destructor.
- ✓ Overloading the = operator.
- ✓ Overloading the << operator to display an object of this class. Declare the overload function as a friend function (Friend Function).

Main	int main() {
	// partie 1
	ListItem* pstr = new StringItem("Hello");
	cout << " Displaying a string element:" << *pstr << endl;
	ListItem* pnbr = new NumberItem(5);
	cout << " Displaying a number element:" << *pnbr << endl;
	// partie 2
	ListItem* pstr_copy = pstr->copy();
	cout << " Displaying a string element after copy: " << *pstr_copy
	<< endl;
	ListItem* pnbr_copy = pnbr->copy();
	cout << " Displaying a string element after a copy: " << *pnbr_copy
	<< endl;
	delete pstr;
	delete pnbr;
	delete pstr_copy;
	delete pnbr_copy;
	// partie 3
	PythonList pyList;
	pyList.addItem(new StringItem("Hello"));
	pyList.addItem(new NumberItem(42));
	pyList.addItem(new StringItem("World"));
	pyList.addItem(new NumberItem(3.14));
	cout << " Original list: " << pyList << endl;
	PythonList copiedList = pyList;
	cout << " Creating a list from another list:" << copiedList<< endl;
	PythonList assignedList;
	assignedList = pyList;
	cout << " Affecting a list to an existing list:" << assignedList << endl;
	return 0;
	}

Display	Displaying a number element: Hello Displaying a number element:5 Displaying a string element after copy: Hello Displaying a string element after a copy: 5 Original list: [Hello, 42, World, 3.14] Creating a list from another list: [Hello, 42, World, 3.14] Affecting a list to an existing list: [Hello, 42, World, 3.14]

	Destructor of PythonList is called. Destructor of PythonList is called. Destructor of PythonList is called.

Part 4:

Internally overload the + operator. The concatenation should produce a new list containing the elements of the two original lists while adhering to best practices for memory management.

The code of the main function must display the following output:

Main	int main() {
	// partie 1
	ListItem* pstr = new StringItem("Hello");
	cout << " Displaying a string element:" << *pstr << endl;
	ListItem* pnbr = new NumberItem(5);
	cout << " Displaying a number element:" << *pnbr << endl;
	// partie 2
	ListItem* pstr_copy = pstr->copy();
	cout << " Displaying a string element after copy: " << *pstr_copy
	<< endl;
	ListItem* pnbr_copy = pnbr->copy();
	cout << " Displaying a string element after a copy: " << *pnbr_copy
	<< endl;
	delete pstr;
	delete pnbr;
	delete pstr_copy;
	delete pnbr_copy;
	// partie 3
	PythonList pyList;
	pyList.addItem(new StringItem("Hello"));
	pyList.addItem(new NumberItem(42));
	pyList.addItem(new StringItem("World"));
	pyList.addItem(new NumberItem(3.14));
	cout << " Original list: " << pyList << endl;
	PythonList copiedList = pyList;
	cout << " Creating a list from another list:" << copiedList<< endl;
	PythonList assignedList;
	assignedList = pyList;

	cout << " Affecting a list to an existing list:" << assignedList << endl;
	//Part 4
	// bonus
	PythonList list1;
	list1.addItem(new StringItem("Hello"));
	list1.addItem(new NumberItem(42));
	PythonList list2;
	list2.addItem(new StringItem("World"));
	list2.addItem(new NumberItem(3.14));
	PythonList concatenatedList = list1 + list2;
	cout << "List 1: " << list1 << endl;
	cout << "List 2: " << list2 << endl;
	cout << "Concatenated List: " << concatenatedList << endl;
	return 0;
	}

Display	Displaying a string element: Hello Displaying a number element:5 Displaying a string element after copy: Hello Displaying a string element after a copy: 5 Original list: [Hello, 42, World, 3.14] Creating a list from another list: [Hello, 42, World, 3.14] Affecting a list to an existing list: [Hello, 42, World, 3.14] List 1: [Hello, 42] List 2: [World, 3.14] Concatenated List: [Hello, 42, World, 3.14] Destructor of PythonList is called. Destructor of PythonList is called. Destructor of PythonList is called. Destructor of PythonList is called. Destructor of PythonList is called. Destructor of PythonList is called.