Task 1: a library project

1. Construct a class *CBook* which includes information about book's title (represented as a string) and its publication year.

2. Implement a member function that prints all the information about a book on a monitor screen.

3. Implement a linked list (*CLibraryList*) of pointers to books. Construct methods that allow:

- check if the list is empty (*IsEmpty*);
- calculate the length of the list (*Length*);
- add a new element to the end of the list (*AddTail*);
- delete an element from the end of the list (*DeleteTail*);
- add a new element to the beginning of the list (*AddHead*);
- delete an element from the beginning of the list (*DeleteHead*);
- consult an element at a given position within the list (*LookAt*);
- delete all the elements from the list (*DeleteAll*).

The list should not have size restrictions. Try to design a generic list of pointers so that you could reuse this class in further projects (use `typedef CBook* DataPtr;`).

4. Implement a global function (*DisplayList*) which receives a pointer to a list and displays all the information about the books stored in the list.

5. Implement a dynamic array (*CLibraryArray*) of pointers to books. Construct methods that allow:

- check if the array is empty (*IsEmpty*);
- calculate the length of the array (*Length*);
- add a new element to the end of the array (*AddTail*);
- delete an element from the end of the array (*DeleteTail*);
- add a new element to the beginning of the array (*AddHead*);
- delete an element from the beginning of the array (*DeleteHead*);
- consult an element at a given position within the array (*LookAt*);
- delete all the elements from the array (*DeleteAll*).

The array should not have size restrictions. Try to design a generic array of pointers so that you could reuse this class in further projects (use `typedef CBook* DataPtr;`).

6. Implement a global function (*DisplayArray*) which receives a pointer to an array and displays all the information about the books stored in the array.

7. Assure that your classes do not have memory leak problems.

8. Test the developed classes with the aid of the following *main* function:

```
int main(int argc, char* argv[])
{
   using namespace std;

   CBook a = CBook("C++", 2014);
   CBook b = CBook("Physics", 1960);
   CBook c = CBook("History", 1934);

   CLibraryList list;
   DisplayList(&list);
   cout << "The list is " << (list.IsEmpty() ? "empty." : "not empty.") << endl;
   list.AddHead(&a); list.AddTail(&c); list.AddHead(&b);
   DisplayList(&list);
   list.DeleteHead(); list.AddHead(&b); DisplayList(&list);
   list.DeleteTail(); list.AddTail(&c); DisplayList(&list);
   cout << "The list has " << list.Length() << " elements" << endl;
   DisplayList(&list);
   list.DeleteAll();
   list.LookAt(8);
   list.AddHead(&a);
   list.DeleteHead();
   list.DeleteTail();

   cout << "--------********************------------" << endl;

   CLibraryArray larray;
   DisplayArray(&larray);
   cout << "The array is " << (larray.IsEmpty() ? "empty." : "not empty.") << endl;
   larray.AddHead(&a); larray.AddTail(&c); larray.AddHead(&b);
   DisplayArray(&larray);
   larray.DeleteHead(); larray.AddHead(&b); DisplayArray(&larray);
   larray.DeleteTail(); larray.AddTail(&c); DisplayArray(&larray);
   cout << "The array has " << larray.Length() << " elements" << endl;
   DisplayArray(&larray);
   larray.DeleteAll();
   larray.LookAt(8);
   larray.AddHead(&a);
   larray.DeleteHead();
   larray.DeleteTail();

   return 0;
}
```

9. Compare the classes *CLibraryList* and *CLibraryArray* in terms of efficiency when adding, deleting and accessing elements. Write a short conclusion.

Task 2:

Imagine that you have a class with the following structure:

```
class CSimple
{
	int m_nSize;
public:
	CSimple  (int nSize) { m_nSize = nSize;  };
	~ CSimple() { } ;
};
```

The *main* function is implemented as follows:

```
int main(int argc, char* argv[])
{
	{
		CSimple  s1;                  //1
		CSimple  s2(2);               //2
		CSimple* ps = new CSimple(3); //3
		delete ps;                    //4
	}                                     //5
	return 0;
}
```

Try to answer the following questions **without using a computer**.

a) Is the code above correct?

b) Which functions are invoked in each of the numbered lines?

Task 3:

What is the output of the following program (**do not use a computer to answer**):

```
void f(int* p)
{
	int k = 1;
	p = &k;
	*p = 100;
}


int main(int argc, char* argv[])
{
	using namespace std;

	int x = 55;
	cout << "x  = " << x << endl;
	int* ptr = &x;
	f(ptr);
	cout << "x  = " << x << endl;
}
```

Task 4: Select true (T) or false (F)

1. One of the supposed benefits of object-oriented programming is code reuse.

2. A C++ variable of type T* can have value 0.

3. If a C++ class' only constructor is declared private, no instances of the class can be created.

4. In C++, all classes derive, directly or indirectly, from the Object class.

5. A class can have two destructors (one for destroying an object from stack and another one for destroying an object from heap).