## Task 1:

1. Construct a class *CPerson* which includes three data members: *age*, *name* e *surname* of a person (*strings*). Implement a function *GetInfo();* which returns a string containing a name and a surname of a person (separated by a space).

2. Construct a class *CProfessor* derived from the class *CPerson*. The class *CProfessor* has an additional attribute which is *m_sArea* – the primary research area. Redefine the *GetInfo();* function so as it returns a research area besides of the name and surname. Do not define the *GetInfo()* function completely from scratch, recur to the function existing in the base class.

3. Construct a class *CStudent* derived from the class *CPerson*. The class *CStudent* has an additional attribute which is *unsigned m_uMecNum* – the student's number. Redefine the *GetInfo();* function so as it returns the student's number besides of the name and surname. Do not define the *GetInfo()* function completely from scratch, recur to the function existing in the base class.

4. Construct a class *CGradStudent* derived from the class *CStudent*. The class *CGradStudent* has an additional attribute which is *CProfessor* m_pProfResp* – a pointer to the supervisor. Redefine the *GetInfo();* function so as it returns the student's name, surname, number as well as supervisor's name, surname and research area. Do not define the *GetInfo()* function completely from scratch, recur to the function existing in the base class.

5. Create a number of objects of types *CPerson*, *CStudent*, *CGradStudent* and *CProfessor* in the *main* function. Call the *GetInfo* function for each of the objects created. Pay attention to the order in which different constructors/destructors are called.

6. Construct a class *CDiscipline* which includes the following members:

```
CProfessor* m_Prof;          // responsible teacher
unsigned m_nNumStudents;     // the number of students
CStudent** m_arStudents;     // array of pointers to students
std::string m_sName;         // discipline's name
```

Suppose that both undergraduate and graduate students may attend a discipline and that the maximum number of students is unlimited. Initially, it is not known how many students will attend a given discipline, therefore the *m_arStudents* array has to grow dynamically with the aid of operators *new* and *delete*.

7. Implement the following *main* function. The global function *void InfoDiscipline (CDiscipline* disc)* prints all the information available about a discipline (its name, teacher's data, the number of students, students' data) on a monitor screen.

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

	CStudent a1 ("Antonio", "Santos", 45, 12345);
	CStudent a2 ("Maria", "Goncalvez", 18, 72456);
	CStudent a3 ("Rita", "Bastos", 29, 12789);
	CStudent a4 ("Jose", "Melo", 34, 13456);
	CStudent a5 ("Manuel", "Brito", 22, 45789);
	CStudent a6 ("Nuno", "Santiago", 24, 45689);
	CStudent a7 ("Nelson", "Rocha", 23, 45189);

	cout << a1.CPerson::GetInfo() << endl;
	cout << a1.GetInfo() << endl;

	CProfessor p1 ("Francisco", "Cardoso", 35, "Programming");
	CDiscipline disc1 ("Object-Oriented Programming", &p1);

	CProfessor p2 ("Jorge", "Matos", 45, "Digital design");
	CDiscipline disc2 ("Advanced Digital Systems", &p2);

	CProfessor p3 ("Fernando", "Marques", 45, "Databases");

	CGradStudent ap1 ("Ricardo", "Almeida", 25, 9189, &p2);
	CGradStudent ap2 ("Pedro", "Cunha", 27, 1089, &p3);

	disc1.AddStudent (&a1);
	disc1.AddStudent (&a2);
	disc1.AddStudent (&a6);
	disc1.AddStudent (&a7);
	disc1.AddStudent (&ap1);

	disc2.AddStudent (&a1);
	disc2.AddStudent (&a2);
	disc2.AddStudent (&a4);
	disc2.AddStudent (&a5);
	disc2.AddStudent (&a6);
	disc2.AddStudent (&ap1);
	disc2.AddStudent (&ap2);

	InfoDiscipline (&disc1);
	InfoDiscipline (&disc2);

	return 0;
}
```

## Task 2:

Is the following code correct? If not, find all the errors.

```
class A
{
      int a1;

public:
      A (int f = 0, int s = 0, int t = 0) : a1(f), a2(s), a3(t) { };
      virtual ~A() {};

      int a2;

protected:
      int a3;
};


class B : public A
{
      int b1, b2, b3;

public:
      B() : A(), b1(0), b2(a2), b3(a3) {  };
      virtual ~B() {};

};
```

---

```
int main(int argc, char* argv[])
{
      B* b = new B();
      return 0;
}
```

## Task 3:

For each of the following statements, indicate whether they are TRUE or FALSE.

1. A class *book* can be derived from the class *library*.
2. A class *vehicle* can include objects of the class *wheel*.
3. When an object of a derived class is created, first of all the constructor of the derived class is executed and after this, the constructor of the base class is executed.