Task 1:

An air company requires a software program to manage its flights. Design and implement in C++ a program that would satisfy the following specifications:

- For each passenger the following information has to be registered: name, surname (*strings*), birth date, sex, and the number of flights completed in the current year. The program should permit to calculate the number of miles that a passenger would receive after completing the flight (provided that the length of the flight is known).

- Implement the following types of passengers:
  - *Economy class* passengers (these are credited with the number of miles which corresponds to the length of the flight).
  - *Business class* passengers (these are credited with the double number of miles).
  - *First class* passengers. These passengers receive the following number of miles (where *n* is the number of flights completed in the current year and *m* is the length of this flight):

$$Miles = \begin{cases} 2 \times m, & if \quad n \le 5 \\ (2 + 0.2 \times n) \times m, & if \quad n > 5 \end{cases}$$

- Assure that your code is extensible and easily modifiable. For instance, it should be easy to add new types of passengers in future, or to alter the way the credited miles are calculated.

- Exemplify the use of the developed classes for a case of a simple program that permits to manage interactively **a single flight** (`flight`). First of all, the user has to specify the length of the flight. Afterwards the program has to permit:

  - to register new passengers;
  - to print the list of all the registered passengers with all the available information about them (name, surname, birth date, sex, type, number of miles to be received during this flight);
  - to display the total number of miles credited on this flight.

- Provide for the possibility of creating a clone of the flight, like:

```
CFlight clone = flight;
```

Task 2:

Analyze the following code fragments and mark lines which contain errors. Explain every error you have found.

```
1. class A
2. {   char* s;
3.   public:
4.      A()
5.      { s = new char[4]; strcpy_s(s, 4, "C++"); }
6.      virtual ~A() { delete [] s; }
7. };
8. int main ()
9. {      A a1;
10.       A a2 = a1;
11.       return 0;
12.  }
```

```
1. class X
2. {      const int m_i;
3.        int a;
4.  public:
5.        X(int I) { m_i = a = I; }
6. };
7.
8. class D : public X
9. {
10.  protected:
11.       D(int d) { a = d; }
12.};
13.
14.int main(int argc, char* argv[])
15.{
16.       D d(3);
17.       return 0;
18.}
```