

---

**Problem 1**

Write a program that encrypts any file and saves the result to another file. The encryption consists of replacing each byte with its “mirror image,” meaning a byte with value  $n$  is replaced by a byte with value  $255 - n$ . Note that this is an involution, i.e., an operation that, when applied to the resulting file, restores the original file (check it!). The names of the input and output files are provided as command-line arguments when running the program.

---

**Problem 2**

Write a program which reads a **binary** file (e.g., an executable *.exe* file, Java *.class* file or a *.pdf* file) and finds words in this file. By word we mean any sequence of at least  $N$  bytes corresponding to ASCII (Latin) letters;  $N$  is by default 4, but may be different. The program writes the words found to a *text* file, each word in a separate line.

For example, taking the *.class* file of the program described above as the input, one can get around 150 lines starting with

```
init
Code
LineNumberTable
main
Ljava
lang
String
StackMapTable
closeResource
Ljava
lang
Throwable
```

Of course, different version of even the same program will give different results.

---

**Problem 3**

Create a class **Person** with fields **name** of type **String**, **birthYear** of type **int** and **car** of type **Car**. Class **Car** should contain fields **make** of type **String** and **color** of type **java.awt.Color**.

Program – in a separate class **Main** – reads a *text* data file using a **BufferedReader**. Data file contains

- One line with a positive integral number (let's call it  $N$ );
- $N$  lines with information about  $N$  **Persons**.

For example

```
7
John  1980 Mercedes 255 255 102
Mary  1997
Alice 1993 Skoda    0 127 153
Sharon 2002 Ferrari 255 0 0
Kenny 1991 Opel    255 255 255
Kate  1999 Mercedes 0 0 255
Jane  2001
```

Each line contains either only the name and birth year of a person, or the name and the birth year of a person, the make of his/her car and its color in the form of three integers from the range  $[0, 255]$  corresponding to three components – red, green and blue. Object of class **Color** (from package *java.awt*) can be then created with the constructor taking three integers, e.g.,

```
Color c = new Color(r,g,b);
```

For each line an object of type **Person** is created (with field **car** equal to **null** if there was no data about this person's car) and inserted into the array of persons.

**Note:** To split a string into “words” separated by any non empty sequence of white characters, you can use the method `str.split("\\s+")` which returns an array of strings – the “words” found in the string `str`.

In class **Person**, create a static function

```
public static Car[] findAllCars(Person[] persons)
```

which takes an array of persons and returns an array (perhaps empty) of all cars owned by them. The array returned must not contain **null** elements.

Write also a static function

```
public static Person[] findOwners(Person[] persons,
                                   String make)
```

taking an array of persons and returning an array (perhaps empty) of those persons who own a car of make **make**. The array returned must not contain **null** elements.

Create a function

```
public static Color findColor(Person[] person,
                               String name, int year)
```

which returns the color of the car belonging to a person with the name **name** and the birth year **year**, or **null** if such a person has not been found or doesn't possess any car.

Write a **main** function which tests all created functions.

---