Programming C/Java Examination

– Exam (**1**) –

Friday, 20 December 2019
Duration: 3 hours

Résumé

This examination has two parts, the first one is to be written in C and the second one in Java. At the end of the exam, you must send your work as a tar archive enclosing source code and text files that answer the examination's questions to `<emmanuel.fleury@u-bordeaux.fr>`.
The archive must conform to the following hierarchy (replace the words enclosed by `<...>` by your name and family name :

```
<FAMILY_NAME>_<Name>-exam/
    +-- heroes_league/
    +-- java/
```

1 C Programming : Heroes League (12 points)

1.1 Project Setting

Knowing that there is no specific dependencies to any library, the goal of this question is to set-up a simple build-system with `make` which will build the final executable program.

Questions

1. Set the complete file hierarchy of the program as follow :

```
<FAMILY_NAME>_<Name>-exam/
    +-- heroes_league/
          +-- heroes_league.c
          `-- Makefile
```

2. Write a `Makefile` with the following targets :
   — `all` : Build the executable `heroes_league` ;
   — `heroes_league` : Build the executable from the source file ;
   — `clean` : Clean-up all unnecessary files and return to a distribution-ready directory.
3. Ensure that you use properly all the usual variables `CFLAGS`, `CPPFLAGS` and `LDFLAGS` in the `Makefile`. And, do not forget to set-up the special target `.PHONY`.

1.2 Main Problem

It is crisis time in North Pole right now. One of the most popular Santa's toys line is the « *Heroes League* ». It is a large set of plastic figures with a *Good* league and an *Evil* league. The problem is that Fizz HollyCakes, the Elf in charge of the whole line, just died of overdose after sniffing too much chocolate powder in a North Pole nightclub (nights are quite long there...). While collecting Fizz's notes about these toys, Santa Claus managed only to recover a lot of paper notes with the hate relationships between some heroes. And, it is not even sure that they are all consistent (as Fizz was a chocolate powder addict which cause some mental disorders on Elves).

Your mission will be to find all the consistent paper notes and discard the inconsistent ones. It is to say that you have to find, given a list of hate links, if yes or no there exists a clear cut into two sets of heroes (Santa will decide which one is the *Good* and which one is the *Evil* afterwards, he is good at it).

— **Input** : The first line of the input gives `T` the number of test cases. `T` test cases follow. Each test case starts with a positive integer `M` on a line by itself – the number of troublesome pairs of Heroes. The next `M` lines each contain a pair of names, separated by a single space.
— **Output** : For each test case, output one line containing `"Case #x: y"`, where `x` is the case number (starting from 1) and `y` is either `"Yes"` or `"No"`, depending on whether the League members mentioned in the input can be split into two groups with neither of the groups containing a troublesome pair.

— **Small Dataset** : $1 \leq \mathtt{T} \leq 100$, $1 \leq \mathtt{M} \leq 10$.
— **Large Dataset** : $1 \leq \mathtt{T} \leq 100$, $1 \leq \mathtt{M} \leq 100$.

Input File

```
2
1
Dead_Bowie Fake_Thomas_Jefferson
3
Dead_Bowie Fake_Thomas_Jefferson
Fake_Thomas_Jefferson Fury_Leika
Fury_Leika Dead_Bowie
```

Output File

```
Case #1: Yes
Case #2: No
```

Figure 1 – Input/Output Example.

Figure 1 shows an example of input and output files. The output display the answers for each test case. Also, more test cases can be found here : `http://www.labri.fr/~fleury/courses/programming/exam/`

Questions

1. Write the file parser that opens an input file and gets the data. Here is the way to use your program :

   ```
   $> ./heroes_league
   heroes_league: error: No input file given !
   $> ./heroes_league heroes_league-xsmall.in
   Case #1: XXX
   Case #2: XXX
   ...
   ```

   If there is no input file, the program must display an error message on `stderr` and return `EXIT_FAILURE`.
2. For the parsing part of the program, we will assume that the input files are all "*conform*", without any syntax error nor semantics error. The only problem that may occur are out of memory errors or a problem on the file (no file given, no read rights, ...).
   To be clear, you need to check the presence of an argument on the command line, then open the file in read (and check that everything went fine), and finally extract each case with all its data.
3. In fact, the main problem is to test bipartiness of the graph given by the hate relationship between heroes. It can be done in linear time with a bi-dimensional array representing the graph. The algorithm is a depth-first search bi-coloring algorithm :
   a. Collect all the links in a bi-dimensional array that represent the graph (hint : it can be safe to assume that the number of links will never be greater than 100).
   b. Color the first node of the first link and label all its neighbors with the other color and recurse.
   c. If you encounter a node already colored, then either it has the right color and you keep going, or it has a different color than expected and the graph is not a bipartite graph.
   d. If the graph is disconnected, then apply the algorithm on each part of it.
4. Write a program that solves the given problem and follows the given specification. Code clarity, code comments and program efficiency will be taken into consideration. Also, be sure that at correct program return, the memory used by the program is properly cleaned.

2 Java Programming (8 points)

Requirements
Question 2.1 must be completed in a plain Unix file.
Answers to 2.2 must be packed into two separate `tar` *archives.*

2.1 General questions

Answer them in a short and clear-cut way.

Questions

1. Casts :
   (a). In the context of subtyping, why *downcasts* are a problem while *upcasts* are not ? Give a concrete example in Java for each of these kinds of casts.
   (b). What are the techniques to deal with downcasts ?
2. What is the main concrete difference in Java between *method overloading* and *method overriding*, and what are their respective uses ?
3. Why does Java need a method selection algorithm ?
4. Why subtyping is expected to induce an *order* relationship ? Indeed, note that :

```
class A extends B {}
class B extends A {}
```

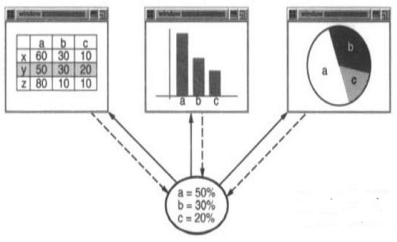
   Generates the following error :

```
Cycle detected: a cycle exists in the type hierarchy between B and A
```

2.2 Problems

Questions

1. *(Factorization of object-oriented code).* In the solution of exercise 4.1 of the “TD” about the simplified aquarium simulation (available in the internet site of the course under “*code source d'une solution d'aquarium*”), the classes implementing the interface `SwimmingStyle` show some code duplication. Modify the code to factorize this duplication in a standard object-oriented way.
2. *(Object-oriented conception).* Design a small object-oriented system for presenting data in a graphical way under various forms. This system should include the notion of *graphical objects*, like spreadsheets, bar charts, scatter plots, etc. and also the notion of *drawable data objects*, i.e., objects containing data in numerical or symbolic forms which can be graphically represented by several graphical objects at the same time. In other words, each drawable data object must be associable with a set of graphical objects :



   Moreover, the system must be such that each time the data of a drawable data object is modified, then all the graphical objects it is associated with must be updated accordingly.
   (a). Use the `Dia` tool to draw a UML class diagram of this graphical system.
   (b). Implement it in Java in a minimal way, by using instances of data object containing integer arrays, associated with two different graphical objects whose graphic capabilities are replaced by plain and minimal writings on the console (i.e. calls to `System.out.print`).