*Università degli Studi di Pisa*

**Laurea Specialistica in Tecnologie Informatiche**

# Advanced Programming

# Mid Term Paper

**Start Date: 29/03/2016**
**Submission deadline: 5/04/2016 (send a single PDF file to attardi@di.unipi.it)**

## Rules:

The paper must be produced personally by the student, signed implicitly via his mail address.

You are allowed to discuss with others the general lines of the problems, provided that each student eventually formulates his own solution. Each student is expected to understand and to be able to explain his solution.

You are allowed to consult documentation from any source, provided that references are mentioned.

It is not considered acceptable:

- **to consult or setup an online forum, to request help of consultants in producing the paper**
- to develop code or pseudo-code with others
- to use code written by others
- to let others use someone"s code
- to show or to examine the work of other students.

Violation of these rules will result in the cancellation of the exam and a report to the Presidente del Consiglio di Corso di Studio.

For the programming exercises you can choose a programming language among C++, C# and Java.

The paper must:
1. be in a single PDF file, formatted readably (**font size ≥ 10 pt** with suitable margins, single column), of **no more than 10 numbered pages**, including code: for each extra page one point will be subtracted from the score.
2. include the student name
3. provide the solution and the code for each exercise separately, referring to the code of other exercises if necessary.
4. cite references to literature or Web pages from where information was taken.

### *Exercise 1*

Given a set of integer intervals $[x_1, x_2]$, with $x_1 < x_2$, write a class that allows finding all intervals intersecting a given interval. The class must provide logarithmic time methods for insertion and removal of an interval, and a method for finding the intersecting intervals. The latter method should be implemented as a generator that computes one result at a time. It is allowed to extend available library classes.
Optional: provide an alternative version implemented using a `yield` operator.

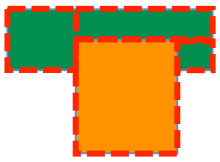
### *Exercise 2*

Consider orthogonal rectangles, defined by the lower left vertex $<x, y>$, and by their width and height, all integer values.
Develop an efficient algorithm for computing the intersections of a set of rectangles, by means of two data structures: one that associates a rectangle to the values of its minimum and maximum $y$ coordinates, ordered by

increasing values and one similar to that of Exercise 1, for detecting rectangles whose width intersects a given interval. Provide a generator that enumerates the pairs of rectangles that have non-empty intersection.

## *Exercise 3*

Given two overlapping rectangles, split them into non-overlapping ones, such as those indicated with a red border in following figure.



Optional: write a program that reads an ordered list by depth of rectangles and produces a tiling of those rectangles by repeatedly splitting those that overlap.

## *Exercise 4*

The yield operator is typically implemented by the compiler introducing a state machine and other transformations to the code. Explain with a simple example the kind of transformations involved.