Università degli Studi di Pisa

Laurea Magistrale in Informatica

Advanced Programming

Middle Term Paper

Start Date: 9/11/2014
Submission deadline: 16/11/2014 (send a single PDF file to attardi@di.unipi.it)

Rules:

The assignment 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 reference material, 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 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.

Exercise 1

Consider a set of tasks among which there is a partial ordering expressing execution constraints between pairs of tasks, i.e. $t_1 < t_2$, means that t_2 must start after t_1 has completed. Define a set of classes to represent such tasks and their precedences and implement a method that checks whether there are cycles in the dependencies.

Exercise 2

Implement an iterator that lists all possible total orderings among tasks that satisfy the constraints. An iterator is a class providing an interface like this:

```
interface Iterator<T> {
    bool  HasNext();
    T Next();
}
```

Exercise 3

Extend the previous classes, in order to represent the duration of a task. Implement a method that finds an ordering with minimum overall execution time and that also computes the minimum number of processors required to perform the tasks according to that ordering.

Exercise 4

Illustrate the constructs of threads in a programming language of your choice. Some programming languages provide an operator `yield`, for implementing a special kind of iterators, called generators. What is the relation between a `yield` method in threads and the `yield` operator in generators?