Università degli Studi di Pisa

Laurea Magistrale in Informatica

Laurea Magistrale in Informatica e Networking

Advanced Programming

Middle Term Paper

Start Date: 26/11/2010
Submission deadline: 6/12/2010 (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 documentation from any source, provided that references are mentioned.

It is not considered acceptable:

- 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.

Exercise 1

Consider a set of tasks among which there is a partial ordering expressing the constraint that a task must precede another. Define a set of classes to represent such tasks and their precedences and 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 2

Extend the previous classes, in order to represent the constraint that a task must complete before the end of another. Implement a method that finds an assignment of tasks to processors which minimizes the number of processors required to perform all tasks while allowing for the maximum parallelism.

Exercise 3

Extend the previous methods and/or classes so that the iterator will check whether there are cycles in the dependencies.

Exercise 4

Discuss and propose how to implement a generic iterator class implemented using threads that provides a `yield(x)` method for returning `x` as a result of a call to `Next()`.

Exercise 5

Illustrate the constructs of threads in a programming language of your choice. What is the relation between a yield method on threads and the yield operator on iterators?