A concurrent binary search tree is a binary search tree on which the three operations  
  Find(key), Insert(key), and Delete(key)  
maybe invoked by concurrent threads.  
(See, e.g., the introductory material of [1] for an explanation of what a binary  
search tree is.)  
  
  
The project consists in  
- implementing a concurrent binary search tree of integer keys  
- providing utilities to test it when at least five threads invoke, in the overall,  
  at least 10 Insert, 5 Delete, and 5 Find  
- providing a short report on your work (to be called REPORT in your submission,  
  and referred to as REPORT below).  
Duplicated keys are not admitted in the search tree, and the tree does not have  
to be rebalanced after insertion and deletion (same assumptions as in [1]).  
  
  
The utilities for testing should provide at least an intelligible way to check what  
the tree is after all (what does it contain? at which leaves?).  
The suggested way to textually represent the tree is to stick to the following  
simple data type written in OCaml syntax (assuming you keep integers also at  
nodes which are not leaves):  
  
  type tree = Empty | Leaf of int | Node of int \* tree \* tree  
  
so that, e.g.,  
                   2  
                  / \  
                 1   3  
  
is rendered by Node (2, Leaf 1, Leaf 3)  
  
and  
                   2  
                    \  
                     4  
                    / \  
                   3   5  
  
is rendered by Node (2, Empty, Node(4, Leaf 3, Leaf 5)).  
  
The output described aboved is considered the default.  
Any further sort of tracing of computation you might wish to implement (e.g. are  
the Find operations successful or not? what is the order in which Insert and  
Delete return?) will be considered a plus.  
Just make it clear in REPORT what the output is supposed to represent.  
  
Providing a graphical output in dot format for the tree is considered a plus  
as well.  
If you choose to do so, please specify it in REPORT.  
  
  
The implementation can fulfill one of three different levels of difficulty,  
corresponding to three different maximal grades to be earned for the evaluation  
of the Project.  
Each student will choose one project out of project A, project B, and project C  
(see below).  
  
  
In every case the project and the relative utilities have to be implemented in  
Java (version 7 or above), and, to guarantee portability, Dott. Gramola (the  
teaching assistant) and I will compile and execute them by plain command lines (javac, java).  
  
  
Every project has to be delivered in the form of a single compressed file containing  
a README.TXT where the author specifies the precise syntax of the command  
lines to use for compilation and execution from the root of the unpacked repository.  
REPORT can be either in txt format or in pdf format, max 4 pages using a font  
size >= 12pt.  
  
  
The submitted compressed file will be named project-type\_matriculation-number  
(e.g. A\_123456789 if you choose the project called project A and  
if your matriculation number is 123456789).  
  
  
The compressed file will be submitted by email  
- by the deadline stated for the relevant exam session (see message posted on  
  April 24th)  
- by the author's unitn email address  
- to the addresses [paola.quaglia@unitn.it](mailto:paola.quaglia@unitn.it) and [lorenzom.gramola@gmail.com](mailto:lorenzom.gramola@gmail.com)  
- with subject KONK\_project-type\_matriculation-number  
  (e.g. KONK\_A\_123456789)  
  
  
The list of projects and associated max grades:  
  
  
PROJECT A (max grade = 18)  
- Implement a course-grained binary search tree  
- Implement test utilities as described above  
- In REPORT describe the locking discipline you used, and explain informally  
  why you believe it works  
  
  
  
PROJECT B (max grade = 24)  
- Implement a fine-grained binary search tree  
- Implement test utilities as described above  
- In REPORT describe the locking discipline you used, and explain informally  
  why you believe it works  
  
  
PROJECT C (max grade = 30)  
- Implement a non-blocking binary search tree  
- Implement test utilities as described above  
- In REPORT describe the main difficulties you encountered, how you solved  
  them, and explain informally why you believe it works  
  
  
[1] Faith Ellen, Panagiota Fatourou, Eric Ruppert, Franck van Breugel:  
Non-blocking binary search trees. ACM PODC 2010: 131-140  
Available at <http://dl.acm.org/citation.cfm?doid=1835698.1835736>  
from within the unitn domain  
(or, equivalently for students abroad or for whichever reason not frequently  
coming at the campus, from a vpn connection to the unitn domain).

EXAM REGULATIONS  
  
  
The exam consists in:  
(i) an oral examination about the topics explained in class as clarified by  
the material uploaded on the didattica-on-line page of the course (Oral, for short)  
(ii) the implementation of a data structure (Project, for short).  
  
Oral and Project contribute 50% to the global evaluation, and both of them have  
to be sufficient in order to pass the exam (i.e. score >= 18/30).  
  
Oral is blocking for Project: Project is to be handed in (and will be marked)  
only after the student passes Oral.  
Also, students are supposed to defend their project after it has been marked.  
  
Oral can be taken at most 3 times out of 5 consecutive exam sessions.  
Project can be handed in more than once only if not sufficient, and in any  
case cannot be submitted more than 3 times out of 5 consecutive exam sessions.  
  
Dates for taking Oral are the official dates of the exam sessions appearing  
on esse3 plus a session, in the form of class-work, on Tue. May 26th at 16:00.  
  
Dates for defending the project are the official dates of the exam sessions  
appearing on esse3.  
Students residing abroad can substitute this by a defense via skype.  
Who wishes to use this possibility will send me an email by her/his unitn-address  
and I will reply with the exact timing the skype call will have to be made.  
  
Strict deadline for handing the project in is 169 hours before the relevant  
exam session as appearing on esse3.  
Further instructions on the way projects should be handed in will be provided  
in the message relative to projects.  
  
Subscription on esse3 is compulsory for both Oral and Project.  
People who intend to take Oral on May 26th will fill in a line at  
<https://docs.google.com/spreadsheets/d/1akGFLkGdYEswp-C0i9YCByzFaJMLi0SFxK5DnoSALkg/edit#gid=0>  
  
Once Oral is successfully done, Project can be handed in any time up to the  
5th exam session for the present edition of the course (February 2016).  
After that session, any partial outcome will be reset.  
  
Official dates of the next three exam sessions are likely to be:  
June 10, 15:00  
July 8, 15:00  
August 19, 15:00