Distributed and Outsourced Software Engineering (DOSE 2015)

Assignment 2: Implementing an AI library

(Mandatory)

Deadline First Iteration: Tuesday, October 27th 5pm (CET)
Final Deadline: Tuesday, November 10th 5pm (CET)

1. Groups

The DOSE project is organized in 10 groups. Each group consists of two teams located in different countries. Each group has to implement one instance of the project, and therefore, the 2 teams have to collaborate and coordinate the development.

The groups have been published at:

https://github.com/DOSE-ETH/dose2015/wiki/Groups

2. Software Development Process

The development of the DOSE project will be done in 2 phases (with 2 different assignments) plus a final phase for testing. The development has to be done using the development process described in *Software Development Process for DOSE 2015.* Each phase is organized in 2 iterations, the deadlines for this Assignment are:

- ***Deadline First Iteration: Tuesday, October 27th 5pm (CET)***
- ***Deadline Second Iteration: Tuesday, November 10th 5pm (CET)***

Task: read the *Software Development Process for DOSE 2015* document, and get familiar with the process.

2.1 Software project leader

Each group has to nominate a *Software Project Leader (SW PL)*. The SW PL is responsible for monitoring, controlling and coordinating the project (for more details see *Software Development Process for DOSE 2015).*

Task: the groups has to decide who is the SW PL and add his/her contact information at: https://github.com/DOSE-ETH/dose2015/wiki/Groups

2.2 GitHub Projects and Forks

2.2.1 Forks

The DOSE project is hosted in GitHub:

https://github.com/DOSE-ETH/dose2015

Each group will fork the dose2015 project, and use it own fork during development (with its own Wiki, issue tracker). After you've created your fork, we will restrict the access to the fork to the students in the group only; you're are not allowed to give access to your fork to other students.

Task: The Software Project Leader has to create the fork, and add a link to the fork in the Wiki https://github.com/DOSE-ETH/dose2015/wiki/Groups

Make sure you enable the issue tracker in your fork. When pushing, creating Wikis, or adding issues to the tracker, please always make sure you're using the fork of your group.

2.2.2 Source code folder for this assignment

Each group has a folder

/02_AI_library/group***i*** where ***i*** is the group number

Each group member must only modify the folder of the group he/she belongs to. This is important, so we can merge the forks later on without conflicts.

Task: In the repository's folder /template_lib, we're providing a project template. The SW PL has to copy the template files to the group's folder (using the group's fork, of course). ***Important: do not modify any class names or routine signatures from the template. They will later be used for automatic testing. Adding more routines is, of course, allowed.***

2.2.3 Issue Tracker

The SW PL, together with the group members, have to plan each iteration with a list of tasks.

Task: Before each iteration, add the tasks (for the upcoming iterations) to the GitHub issue tracker. Assign who is responsible for each task. Tag the tasks of different iterations with the following labels:

lib-iteration1, ***lib-iteration2***: tasks in Assignment 2

app-iteration1, ***app-iteration2***: tasks in Assignment 3

2.2.4 Wiki

After each iteration, the SW PL has to create a Wiki page (called Iteration Report):

- Number of open and closed issues
- Number of new additional tasks / issues created in the iteration
- Number of tasks / issues closed during the iteration
- For each group member:
  - Number of tasks closed by the group member
  - Number of open issues assigned to the group member

Task: After each iteration, add a link to the Iteration Report in the DOSE Wiki page:

https://github.com/DOSE-ETH/dose2015/wiki/Groups

3. Learn Eiffel & Install EiffelStudio

3.1 Learn Eiffel

We've prepared an *Introduction to Eiffel* which explains all important Eiffel language constructs. The introduction consists of 3 slide sets:

1. http://se.inf.ethz.ch/courses/2015b_fall/dsl/exercises/intro_eiffel_part1.pdf
2. http://se.inf.ethz.ch/courses/2015b_fall/dsl/exercises/intro_eiffel_part2.pdf
3. http://se.inf.ethz.ch/courses/2015b_fall/dsl/exercises/intro_eiffel_part3.pdf

Task: each student must work through the 3 slide sets. Complete the hands-on exercises (there are 9 in total). Submit your solution to each hands-on exercise in Codeboard (make sure you log in before writing a solution, click the "Submit" button to submit your solution).

3.1 Install EiffelStudio

We've prepared instruction on how to install EiffelStudio:

https://github.com/DOSE-ETH/dose2015/wiki/Installing-EiffelStudio

Task: each student must install EiffelStudio 15.08 on their computer.

4. Project

The topic of the project for this assignment is a library for AI search algorithms, including for example the following search algorithms:

- uninformed single agent (depth-first search, breadth-first search, iterative deepening, bounded DFS and BFS, etc.)
- heuristic single agent (hill climbing, best first search, etc.)
- adversary search (minimax, bounded minimax, minimax with alpha-beta pruning, negascout, etc.)

Your group will have to self-organize how you split the workload between the all team members.

Task: your group has to implement the AI library. The project template has a skeleton for all algorithms that have to be implemented. Below, is the list of algorithms with their priorities. *Priority 1 ("Mandatory")* are mandatory and must be implemented to successfully finish the DOSE project; *Priority 2 ("Nice to have")* are algorithm *nice to have* and at least 50% of these algorithms must be implemented to successfully finish the DOSE project; *Priority 3 ("Optional")* are optional algorithms.

Priority 1	Priority 2	Priority 3
- Bounded depth first search - Bounded breadth first search - Best first search - Hill climbing - Minimax	- Unbounded depth first search with cycle checking - Lowest-cost first search - Heuristic depth first search - Steepest ascent hill climbing - Minimax with alpha beta pruning	- A* Search - Iterative deepening - Principal variation search (NegaScout)