OLYMPIA TEACHER USER GUIDE

Create Questions - Programming Question Type (Detailed Version)

A. As A Teacher

1. Create programming question.
    1.1. Go to Course Page.
    1.2. Click “Question bank” under “Course administration” to go to Question Bank Page.
    1.3. Click “Create a new question …” button.
    1.4. Select “Programming” from the list of question types.
    1.5. Click “Add” button.
    1.6. Note for “General”:
        1.6.1. "Default mark" is the default maximum mark for the question. "Default mark" must be filled by the total of "Blackbox score" and/or "Whitebox score", there will be error if "Default mark" is different with total score.
        1.6.2. "Categories" is the categories for the question. Multiple categories could be given as comma (",", ASCII code 44) separated string.
        1.6.3. "Difficulty" is the difficulty for the question. Difficulty could be given as integer.
        1.6.4. "Tries limit" is the tries limit for the question. The minimum tries limit is 1 time. The number of tries for the question depends on “Question behaviour”.
        1.6.5. You must upload a zip file in “Grader files” which contains all of the files needed for grading.
            1.6.5.1. Please verify that file name filled in any of the field actually exists, because there will be error if any of the file doesn’t exist.
            1.6.5.2. Be careful not to put any of the file inside folder in the zip file that will be uploaded.
    1.7. Note for “Blackbox”:
        1.7.1. “Time limit” is the allotted time in milliseconds for program execution. The minimum time limit is 100 milliseconds.
        1.7.2. “Memory limit” is the allotted memory in kilobytes for program execution. The minimum memory limit is 8192 kilobytes.
        1.7.3. “Compile error score” is the score that will be given to the student if the program got “Compile error". "Compile error score" will not be given if the evaluator got "Evaluator error".
        1.7.4. “Builder” is the builder that will be used to create an executable file. Builder file (i.e. makefile, build.xml, pom.xml, build.gradle, build.sh) must be provided in “Grader files” or “Answer files”. You must specify "Compile" and "Run" under “Blackbox Languages” for every language. Currently, supported “Builder” is makefile.
        1.7.5. “Exact" and "Tolerant” are evaluator types.
            1.7.5.1. Select “Default Evaluator” if you want to use default evaluator. The default evaluator for "Exact" is using standard string comparison. The default evaluator for "Tolerant" is using standard string comparison with all whitespace trimmed.

1.7.5.2. Select “Custom Evaluator", if you want to use custom evaluator. Evaluator file must be provided in "Grader files". You must specify the "Evaluator" file name and language.

1.7.5.3. You must specify "Exact score" and/or "Tolerant score" for every "Test case" under "Blackbox Test Cases".

1.8. Note for “Blackbox Languages”:

1.8.1. Check the checkbox for every language allowed to answer the question.

1.8.2. "Compile" is the rule (makefile), target (ant, maven), task (gradle), or functions (shell) to build executable file.

1.8.3. "Run" is the executable file to run resulted from build.

1.8.4. Note for "Plain":

1.8.4.1. "Plain" language can not be used with another language simultaneously.

1.8.4.2. "Answer files" must contains answer (.ans) file.

1.8.4.3. The answer file must have the same name as the output file.

1.8.4.4. If the answer file does not exist for some output file, it will be considered as "Wrong answer".

1.9. Note for “Blackbox Test Cases”:

1.9.1. Manual test cases configuration:

1.9.1.1. “Input” is the input file name. “Output” is the output file name. "Input" and "Output" must not be empty.

1.9.1.2. “Exact score” and "Tolerant score" are the score of corresponding “Input” and “Output” pair.

1.9.1.2.1. If “Exact" evaluator is "None", the “Exact score” will be ignored. If “Exact" evaluator is not "None", the “Exact score” must not be empty.

1.9.1.2.2. If “Tolerant" evaluator is "None", the “Tolerant score” will be ignored. If “Tolerant" evaluator is not "None", the “Tolerant score” must not be empty.

1.9.1.2.3. "Blackbox score" will be determined by sum of “Exact score” and/or “Tolerant score”.

1.9.1.3. If any of “Input”, “Output”, “Exact score”, or "Tolerant score" is invalid, the test case will be ignored.

1.9.1.4. If you want to specify more test cases, click "Blanks for 5 more test cases" button.

1.9.2. Automatic test cases configuration:

1.9.2.1. You must leave all fields under “Blackbox Test Cases” empty.

1.9.2.2. Test cases must consists of input (.in) and output (.out) file.

1.9.2.3. Test cases could consists of exact (.exact) and/or tolerant (.tolerant) file, there are 3 possibilities:

1.9.2.3.1. The score for each test case will be taken from exact and tolerant file.

1.9.2.3.1.1. If “Exact" evaluator is "None", the exact file will be ignored. If “Exact" evaluator is not "None", the exact file must exists.

1.9.2.3.1.2. If “Tolerant" evaluator is "None", the tolerant file will be ignored. If “Tolerant" evaluator is not "None", the tolerant file must exists.
1.9.2.3.1.3. "Blackbox score" will be determined by sum of exact and/or tolerant file.
1.9.2.3.2. The score for each test case will be prorated.
1.9.2.3.2.1. If “Exact" evaluator is not "None", the exact file must not exists.
1.9.2.3.2.2. If “Tolerant" evaluator is not "None", the tolerant file must not exists.
1.9.2.3.2.3. "Blackbox score" will be determined by subtracting "Whitebox score" for corresponding language from "Default mark".
1.9.2.3.2.4. Let X be "Blackbox score" divided by the number of evaluator which is not "None". X will be rounded down to the nearest integer and the remainder will be added to the last evaluator.
1.9.2.3.2.5. Let Y be X divided by the number of input and output pair. Y will be rounded down to the nearest integer and the remainder will be added to the last input and output pair.
1.9.2.3.3. The score for each test case can not be determined (i.e. the score file for evaluator which is not "None" some does exist, some does not exist).
1.9.2.4. Test cases must exists in either "Grader files" or "Answer files", there are 2 possibilities:
1.9.2.4.1. If there is at least one valid test cases in "Grader files", test cases configuration will be taken from "Grader files".
1.9.2.4.2. If there is no valid test cases in "Grader files", test cases configuration will be taken from "Answer files".

1.10. Note for “Whitebox”:
1.10.1. Select “No operation checker” if you want to use checker which does nothing.
1.10.2. Select “Custom checker", if you want to use custom checker. Checker file must be provided in "Grader files". You must specify the "Checker" file name and language.
1.10.3. "Config" is the configuration file name of the corresponding checker.
1.10.4. "Language" is the language for which the corresponding checker will be applied.
1.10.5. "Score" is the score of the corresponding checker.
1.10.6. "Whitebox score" will be determined by maximum of sum of score for each checker for each language. Be advised that sum of score for each checker could be different for different language.
1.11. Click “Save changes” button.
1.12. Note for error saving programming question:

1.12.1. You should read the error messages and edit the question accordingly.
1.12.2. You should never use the question if there exists any error, because it may not behave as expected.
1.12.3. Click "Continue" button to continue editing.

1.13. Note for blackbox grading process:
1.13.1. There are 3 phases (i.e. compile, execute, evaluate) for grading process:
1.13.1.1. If the program fails in compile phase, the verdict will be "Compile error".
1.13.1.2. If the program fails in execute phase, the verdict will be "Runtime error", "Memory limit exceeded", or "Time limit exceeded".
1.13.1.3. If the program fails in evaluate phase, the verdict will be "Wrong answer".
1.13.1.4. If the program passes all 3 phases, the verdict will be "Accepted".
1.13.1.5. Execute phase depends on compile phase. If the program fails in compile phase, grading process will not proceed to execute phase.
1.13.1.6. Evaluate phase depends on execute phase. If the program fails in execute phase, grading process will not proceed to evaluate phase.
1.13.2. All 3 phases will execute commands in a sandbox to do compile, execute, and evaluate. If one of the following constraints violated during execution, the program considered failed in corresponding phase.
1.13.2.1. Maximum number of processes is 50.
1.13.2.2. Maximum file size that can be created is 102400 KB.
1.13.2.3. Maximum number of open files is 50.
1.13.3. "Compile error" applied to both compiled language (compile will be performed) and interpreted language (compile / lint will be performed). The compile process must be done within 30000 milliseconds.
1.13.4. The program must exit with exit code 0. If the program exit with non-zero exit code, the verdict will be one of the following:
1.13.4.1. Exit code 124 will got "Time limit exceeded".
1.13.4.2. Exit code 9 or 137 will got "Memory limit exceeded".
1.13.4.3. Other exit code will got "Runtime error".
1.13.5. If you use custom evaluator and the program fails, the verdict will be "Evaluator error".

1.14. Note for blackbox custom evaluator and whitebox custom checker:
1.14.1. The evaluator or checker could fail because one of the following reasons.
1.14.1.1. The evaluator or checker got "Compile error".
1.14.1.2. The evaluator or checker got "Runtime error".

1.14.1.3. The evaluator or checker got "Memory limit exceeded". The memory limit is 262144 kilobytes.
1.14.1.4. The evaluator or checker got "Time limit exceeded". The time limit is 10000 milliseconds.
1.14.1.5. The evaluator does not output "FAILED" or "PASSED".
1.14.1.6. The checker does not output a valid score.
1.14.2. Blackbox custom evaluator input and output:
1.14.2.1. The evaluator will be given 3 command-line arguments (i.e. teacher input file name, teacher output file name, student output file name).
1.14.2.2. The first line of the evaluator output must be "FAILED" (if the program considered failed in evaluate phase) or "PASSED" (if the program considered passed in evaluate phase).
1.14.2.3. The next line of the evaluator output until EOF (End Of File) will be considered as description.
1.14.3. Whitebox custom checker input and output:
1.14.3.1. The checker will be given arbitrary number of command-line arguments (i.e. the configuration file name of the corresponding checker and the file name of the files which are part of the program).
1.14.3.2. The first line of the checker output must be a valid score. A valid score could be a real number. A valid score must be greater than or equal to 0 and lower than or equal to the score of the corresponding checker.
1.14.3.3. The next line of the checker output until EOF (End Of File) will be considered as description.
1.14.4. Be careful with different behaviour of different language when parsing command-line arguments (e.g. in C, C++, Pascal, PHP, Python the command that was used to run the program is included whereas this not the case in Java).

2. Create quiz.
2.1. Go to Course Page.
2.2. Click “Turn editing on” button.
2.3. Click “Add an activity or resource”.
2.4. Select “Quiz” activity.
2.5. Click “Add” button.
2.6. Fill required fields.
2.7. Note for “Question behaviour”:
2.7.1. The behaviour of programming question determined by “How questions behave”, there are 2 possibilities:
2.7.1.1. If “Deferred feedback” or “Deferred feedback with CBM” is selected, then student will not get any feedback during the attempt and student can only try the question once during the attempt.
2.7.1.2. If “Adaptive mode” or “Adaptive mode (no penalties)” or “Immediate feedback” or “Immediate feedback with CBM” or

"Interactive with multiple tries" is selected, then student will get feedback immediately and student can try the question multiple times (if student's answer not yet correct for all test cases) depending on the tries limit for the question.

2.8. Note for "Review options":
  2.8.1. There are 4 options that control when the students can see the review of Quiz they attempted:
    2.8.1.1. "During the attempt" means the review chosen will appear while the students are taking the quiz attempt.
    2.8.1.2. "Immediately after the attempt" means the reviews chosen will appear within 2 minutes of the attempt being finished (when the student hits the 'Submit' button).
    2.8.1.3. "Later while the quiz is still open" means the reviews chosen will appear after the 2 minutes are up, and before the quiz close date.
    2.8.1.4. "After the quiz is closed" means the review chosen will appear after the quiz close date has passed. If the quiz does not have a close date, this state is never reached.
  2.8.2. There are 7 options for what review the students will see at the specified time (point 2.8.1.):
    2.8.2.1. "The attempt" shows the questions in the quiz.
    2.8.2.2. "Whether correct" shows the current state of the question ("Tries remaining", "Correct", "Partially correct", "Incorrect", "Requires grading", "Complete", "Incomplete answer", "Not answered", etc.).
    2.8.2.3. "Marks" shows marks for the question in the quiz if the student can see both "The attempt" and "Marks" and marks for the quiz.
    2.8.2.4. "Specific feedback" shows the feedback for each answer for the question, based on the answer from the student.
    2.8.2.5. "General feedback" shows the feedback for each answer for the question.
    2.8.2.6. "Right answer" shows correct answer for the question.
    2.8.2.7. "Overall feedback" shows the feedback for the quiz, based on the student's marks for the quiz.
2.9. Note for "Overall feedback":
  2.9.1. Under the "Overall feedback" there are 2 types of components:
    2.9.1.1. "Grade boundary" shows the boundary where the feedback specified will be shown.
    2.9.1.2. "Feedback" shows the feedback that will be shown for the specified boundary.
2.10. Click "Save and display" button.

3. Add question to quiz.
  3.1. Go to Course Page.
  3.2. Click Quiz name to go to Quiz Page.
  3.3. Click "Edit quiz" under "Quiz administration".

3.4. Click “Add”, “from question bank”.
3.5. Select question that you want to add.
3.6. Click “Add selected questions to the quiz” button.
3.7. Fill “Maximum grade” for the quiz located on the top right and “Maximum mark” for each questions right beside the Question Name and Description.
3.8. Note for “Maximum grade” and “Maximum mark”:
3.8.1. When the question added to the quiz, “Maximum mark” for the question will be taken from “Default mark” for the question.
3.8.2. When the “Maximum mark” for the question changed, only that question will be affected, the "Default mark" for the question will not be affected.
3.8.3. When the “Default mark” for the question changed, only the “Maximum mark” for the question after the change is made will be affected, the “Maximum mark” for the question before the change is made will not be affected.
3.9. Click “Save” button.

4. View quiz grade.
4.1. Go to Course Page.
4.2. Click Quiz name to go to Quiz Page.
4.3. Click “Results” under “Quiz administration”.

5. Regrade quiz attempt.
5.1. Go to Course Page.
5.2. Click Quiz name to go to Quiz Page.
5.3. Click “Results” under “Quiz administration”.
5.4. Note for quiz which contains programming question:
5.4.1. You may see grade changes after “Regrade completed successfully” message displayed. This is because grading process might still be running.
5.4.1.1. For “Regrade all”, “Regrade attempts marked as needing regrading”, “Regrade selected attempts” you could check all attempts state. If all attempts not in requires grading state, then regrade actually completed.
5.4.1.2. For “Dry run a full regrade” you don’t know when regrade actually completed.
5.4.2. The result of “Dry run a full regrade” would be slightly different:
5.4.2.1. Attempts marked as needing regrading could contain attempts which actually doesn’t need to be regraded.
5.4.2.2. Attempts which actually needs to be regraded will be marked as needing regrading.
5.5. Note for “Regrade all”:
5.5.1. If you click this button, all attempts will be regraded.
5.6. Note for “Dry run a full regrade”:
5.6.1. If you click this button, you can see all attempts which grade would change if you did a regrade (the attempts will be marked as needing regrading).
5.7. Note for “Regrade attempts marked as needing regrading”:

5.7.1. You may not see this button if there are no attempts marked as needing regrading.
5.7.2. If you click this button, all attempts marked as needing regrading will be regraded.
5.8. Note for “Regrade selected attempts”:
5.8.1. If you click this button, all selected attempts will be regraded.

6. View report.
6.1. Go to Course Page.
6.2. Click “Grades” under “Course administration”.