Integrating Parsons Puzzle into MyUni Design Document

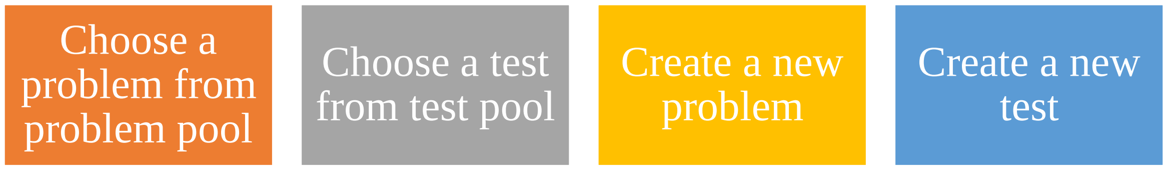
PROJECT GOALS

The objective of this project is to design and develop a universal design of parson's puzzle and implement it into customised H5P, that will transform the process of learning for students. The project will include the following goals:

- Gather information, research and design a universal parsons puzzle which can be implemented for the subjects where algorithmic thinking is required.
- Development of environment settings.
- Implementation of parson's puzzle H5P content.
- Integrating H5P with Drupal, further Canvas.

Each instructor will have access to:

1. Problem Pool: A list of problems created by Instructors. It's hard to build a frame to create and design the problem. Each puzzle should be designed well by the instructor and to make sure that it can be repeatedly used, we create a problem pool.
2. Test Pool: A list of tests having problems created by Instructors. Same as the problem pool.
3. Option to create a new test.
4. Option to create a new problem.


Choose a problem from problem pool
Choose a test from test pool
Create a new problem
Create a new test

Interface for instructors will include the following options:

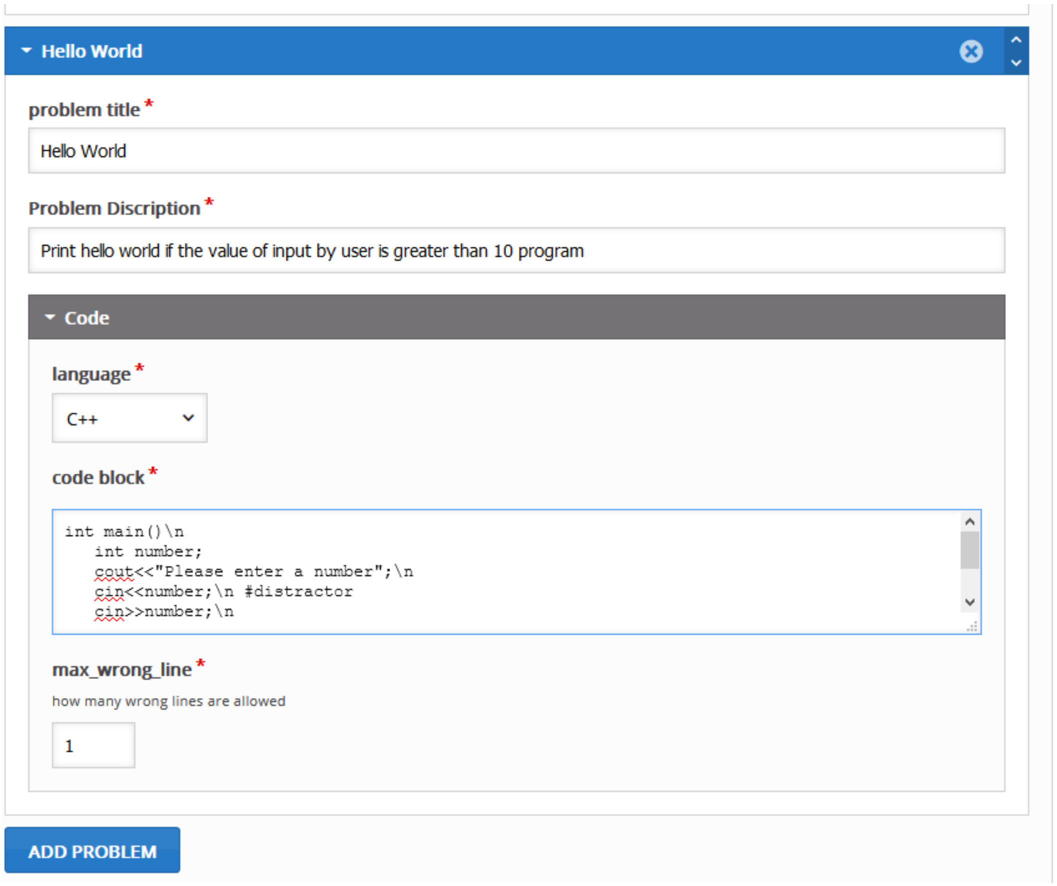
1. My tests: The tests created by the instructor.
2. My questions: The questions created by the Instructor.

The instructor can choose a test from the test pool or the problem from the problem pool. According to:

1. Tags – difficulty or topics.
2. Used times(frequency): used to evaluate the quality of the puzzle which are being re-used by other assessors/ the instructor who created it can get the credits for creating the test
3. Review (Rating star): rating star or comments from both the instructor and the students for the tests used by the instructors and the tests attempted by the students.

Instructor can create a new problem or test with the following options/rules:

1. Predefined syntax: Instructors can use pre-defined syntax rule to create a new problem in a coding block (with the line number of the codes):
   #distractor: distractor for increasing the difficulty for the parsons puzzle.
   Example: a = 2 **#distractor**; automatically generated the distractor by using the predefined syntax.


Hello World

problem title *

Hello World

Problem Discription *

Print hello world if the value of input by user is greater than 10 program

Code

language *

C++

code block *

```
int main()\n
   int number;
   cout<<"Please enter a number";\n
   cin<<number;\n #distractor
   cin>>number;\n
```

max_wrong_line *

how many wrong lines are allowed

1

ADD PROBLEM

Figure 1: The instructor using the predefined syntax, #distractor for our project.

As there are two types of distractor, paired and un-paired, further divisions can be made within the syntax rule and the #d# can be kept default distractor which creates a distractor at the line in which the syntax rule is being used.
a = 100 **#unpaireddistractor**; creating an unpaired distractor
a = 100 **#paireddistractor#**; creating a paired distractor

#fs# #fe#: for the adding the fill in the blank in between the drag and drop, the following syntax rule can be used to create a blank to be filled by the student while attempting the problem
Example: a = **#fs#** 2 **#fe#**; Display: a = ____ (the answer is 2)
In between the rule, lies the answer and the student should type the same answer in the blank while attempting the puzzle to get it correct.

#t#: indentation
Example: #ttt# It's an example of 3 indentation as one t would define one level of indentation in the code required.

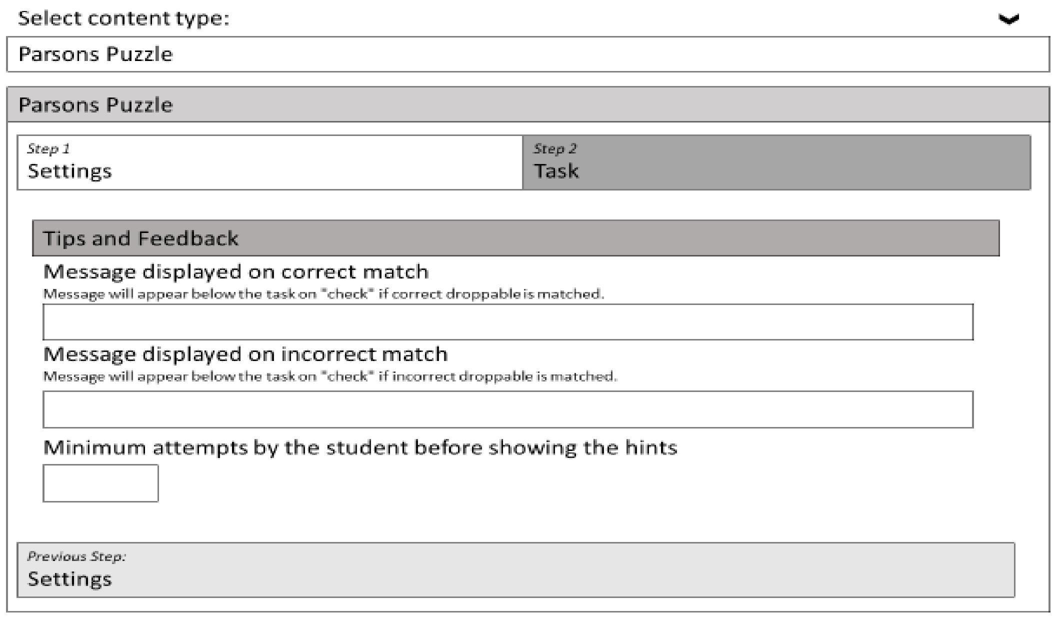
#c#: comment line
Comments will be given whenever required by the instructor in between the code and the Instructor can use the syntax rule to generate comments in between the problem.
Example: #c# assume the value of a to be -19.

2. Expected Actions: The instructor can set the expected actions of students to solve drag and drop (total number of actions within which the student is required to complete the problem). This option is necessary to prevent over-attempts of actions. Students can use hit and trial without actually knowing how to solve the problem, hence this action will prevent students from mindlessly working on the quiz as well as prevent the chances of hit and trial from students.

3. Hints and Support: Instructor can write the necessary hints if required for the problem and even add the details of the learning objectives of the puzzle for the students to get a better idea of the problems objectives to help them relate it to the course material they might be reading before attempting the puzzle. The instructor can also add activity diagrams or flow chart.
   For making students to think and attempt the problem, there will be an option for setting the minimum try times before showing the hints, so that students first attempt the problem on their own before looking out for hints. It is optional and depends on the instructor.

   Programming is about the whole big picture. Each line of codes effects the other. It's a system. If in feedback I simply get the information of which line has error is not helpful because we still need to think why it's wrong. For example, in a two-dimensional array, we usually use nested for-loop to iterate it. If we are given a puzzle to reorder or organize the for-loop, I probably would make a mistake. But if I only get a feedback of which line incurs the error, it's probably not helpful. Instructor should provide a 2D diagram of how does 2-dimension array look alike, help the students generate a picture in their head.

```
    i0 i1 i2 i3
j0: 1  2  3  4
j1: 2  3  4  5
j2: 4  5  6  7    --- 2 D array visualization
```



Select content type:

Parsons Puzzle

Parsons Puzzle

Step 1 Settings | Step 2 Task

Tips and Feedback

Message displayed on correct match
Message will appear below the task on "check" if correct droppable is matched.

Message displayed on incorrect match
Message will appear below the task on "check" if incorrect droppable is matched.

Minimum attempts by the student before showing the hints

Previous Step: Settings

Figure 2: Instructor can add tips or feedback whenever necessary for the requires test/problem.

4. Time limit: Instructor can choose to create a timed problem/test by setting the time limit for it.

5. Tag the difficulty level of the test/problem: The instructor tags the difficulty and topic of the problem as per the requirement.
   Options for difficulty: Easy, Medium, Hard.
   Options for topics: Create a new topic or choose from the already created topics.

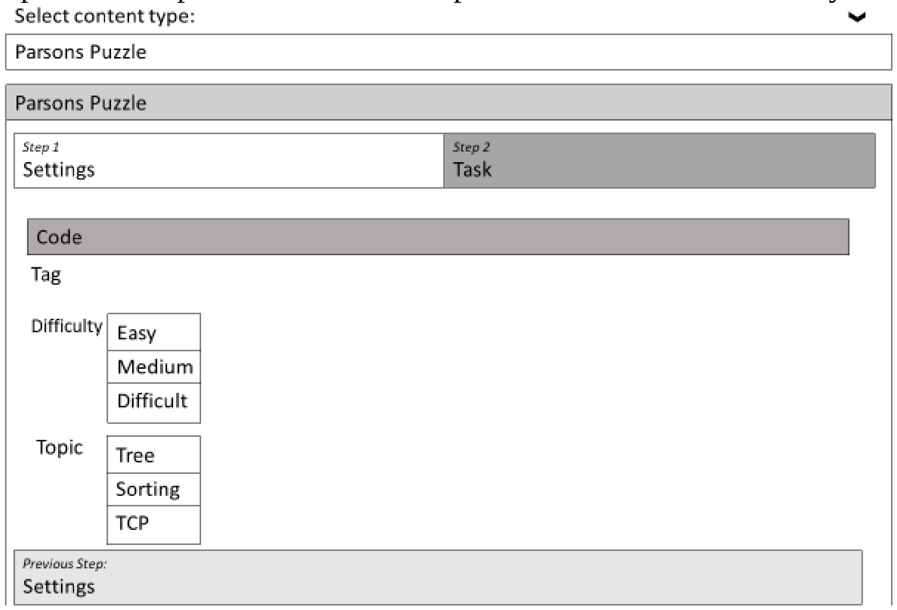

Select content type:
Parsons Puzzle
Parsons Puzzle
Step 1 Settings
Step 2 Task
Code
Tag
Difficulty: Easy / Medium / Difficult
Topic: Tree / Sorting / TCP
Previous Step: Settings

   Figure 3: Instructor can choose the difficulty and topic. If its not present, instructor can add new topic into the topic list.

6. Due date of the problem/test: The instructor can choose to add the due date of the problem or test by which the students will have to complete the it. This will make sure that the students are working over the problem/test continuously as per the requirements of the design of the course undertaken by the instructor.

7. Allowed attempts of the problem/test: The instructor can choose to set the total attempts which are allowed for the problem/test.

8. Grade/Marks: The instructor can add the range of marks for passing or failing the problem/test. The instructor can design the marking as per the problem/test.

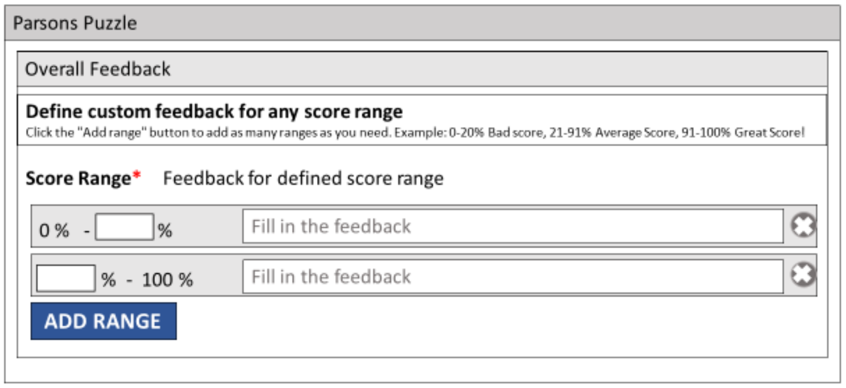

Parsons Puzzle
Overall Feedback
Define custom feedback for any score range
Click the "Add range" button to add as many ranges as you need. Example: 0-20% Bad score, 21-91% Average Score, 91-100% Great Score!
Score Range* Feedback for defined score range
0 % - ___ % | Fill in the feedback
___ % - 100 % | Fill in the feedback
ADD RANGE

   Figure 4: Instructor can define the range of the marking and the feedback for that range for the students attempting the test/problem.

After the students attempt the test/problem:

9. Availability of the answer of the test/problem: The instructor can choose to release the answers of test/problem. Instructor can choose to release the answer immediately after the student attempts the test/problem or after all the students have attempted the test/problem or after the due date of the test/problem. The instructor can also choose not to release the answers if the quiz has to be attempted in the next years as well. A panel to show the solution to the students is desirable.

10. Availability of the grades: The instructor can choose to post the grades for the test/problem the student attempted by just posting the grades after the attempt or posting the grades after all the students have attempted the test/problem.

11. Availability of the mistakes committed by the student: The instructor can choose to show the mistakes committed by the student after the student attempts the quiz. The instructor can also set the limit after which the mistakes of the student will be available to prevent the over-use of this feature or to prevent the hit and trial by the students.

12. The option to group the students who are progressing alike or are having difficulty in the same topics or are making similar mistakes.

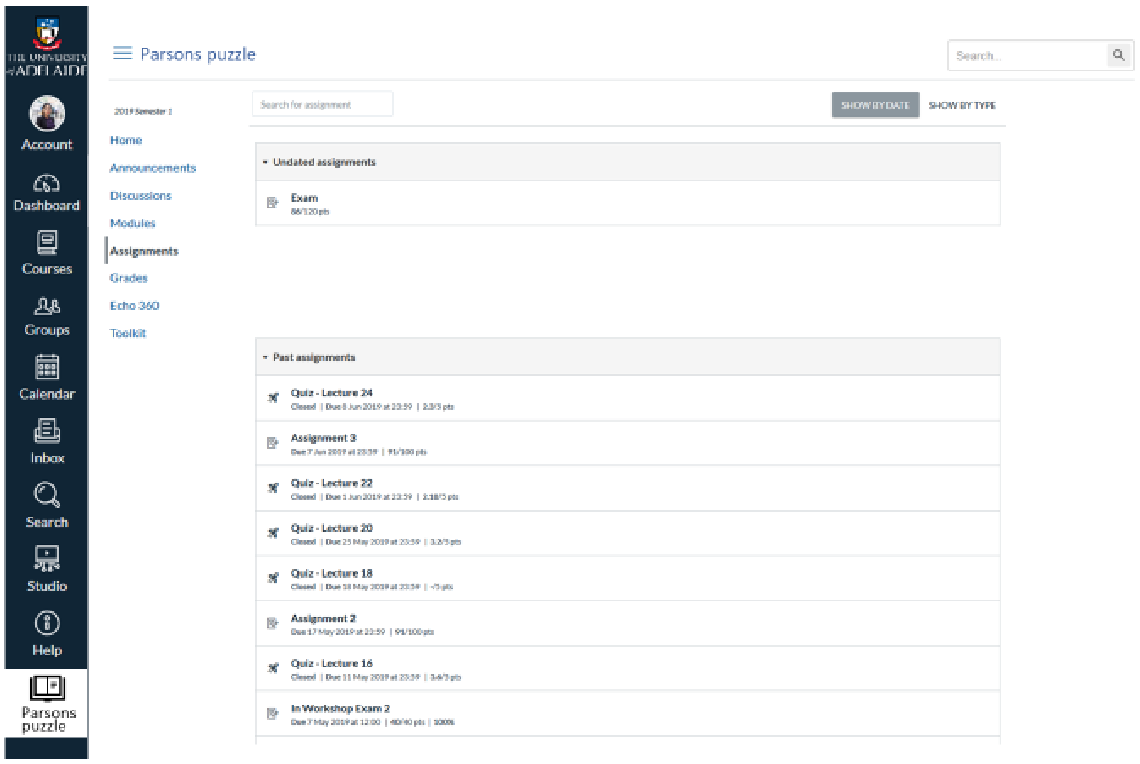
13. Create supplement problem/test for a specific group of students if they fail the test/problem.

The data generated after the student attempts the test/problem for the instructor:

1. The number of students who take the test/solve the problem.
2. The grades of the students who attempted the test/problem.
3. The normal distribution of time spent by the student to solve the problem/test.
4. The normal distribution of try times (the number of times the test/problem was attempted by student).
5. The normal distribution of the average number of actions by the student to solve the problem/test.
6. The occurrence of hints for a student while solving the test/problem.
7. The date and time at which the students attempted the quiz.
8. The pass ratio for every try by the student.
9. The reviews given the student after attempting the problem/test each time. It will be a rating star, comments (any problem or difficulty while attempting the quiz or if the student wants some extra information to attempt the quiz) and the difficulty experienced by the student.

Interface for students will include the following options:

1. Track their own progress by getting the option of assignments published and sorted as per the due date of the assignments. If there is no due date they appear according to the date published by the instructor under the undated assignments.
2. Attempt history: The attempt number, time takes and the grade for the problem/test taken by the student. If exact grade has not to be released, student will receive the pass or fail grade.
3. Students can even check the type of questions they are performing correct.



Undated assignments

Exam
86/120 pts

Past assignments

Quiz - Lecture 24
Closed | Due 8 Jun 2019 at 23:59 | 2.3/5 pts

Assignment 3
Due 7 Jun 2019 at 23:59 | 91/100 pts

Quiz - Lecture 22
Closed | Due 1 Jun 2019 at 23:59 | 2.18/5 pts

Quiz - Lecture 20
Closed | Due 25 May 2019 at 23:59 | 3.2/5 pts

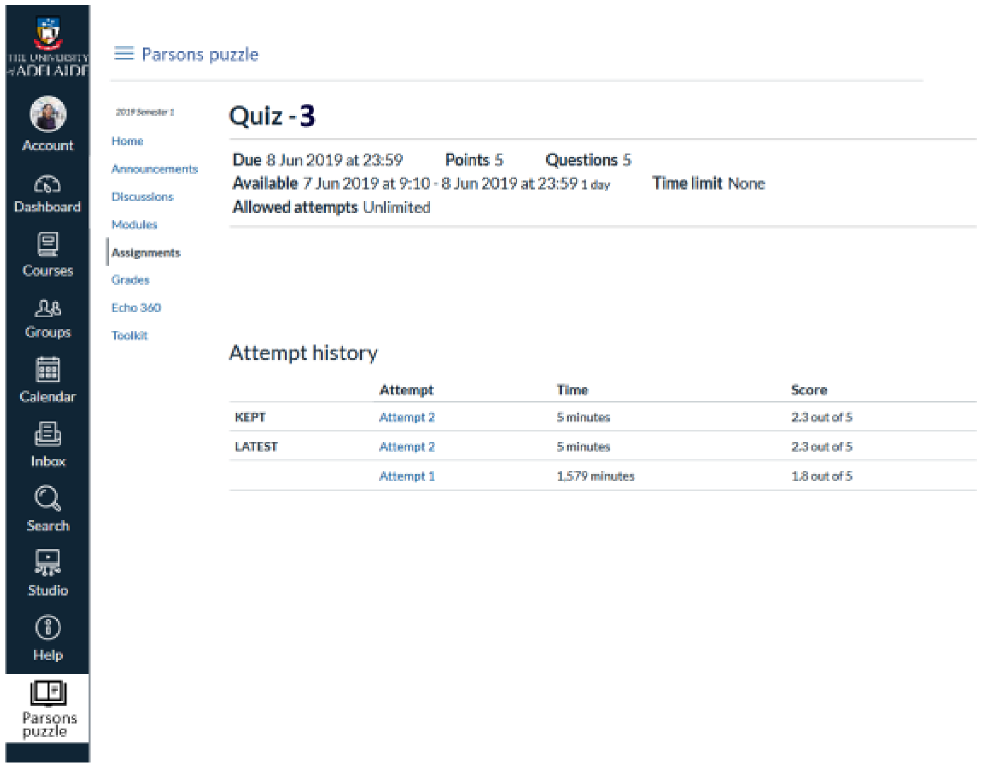
Quiz - Lecture 18
Closed | Due 18 May 2019 at 23:59 | -/5 pts

Assignment 2
Due 17 May 2019 at 23:59 | 91/100 pts

Quiz - Lecture 16
Closed | Due 11 May 2019 at 23:59 | 3.6/5 pts

In Workshop Exam 2
Due 7 May 2019 at 12:00 | 40/40 pts | 100%

Figure a: Parsons puzzle function integrated to MyUni, will show the undated assignments, past assignments updated and upcoming assignments.



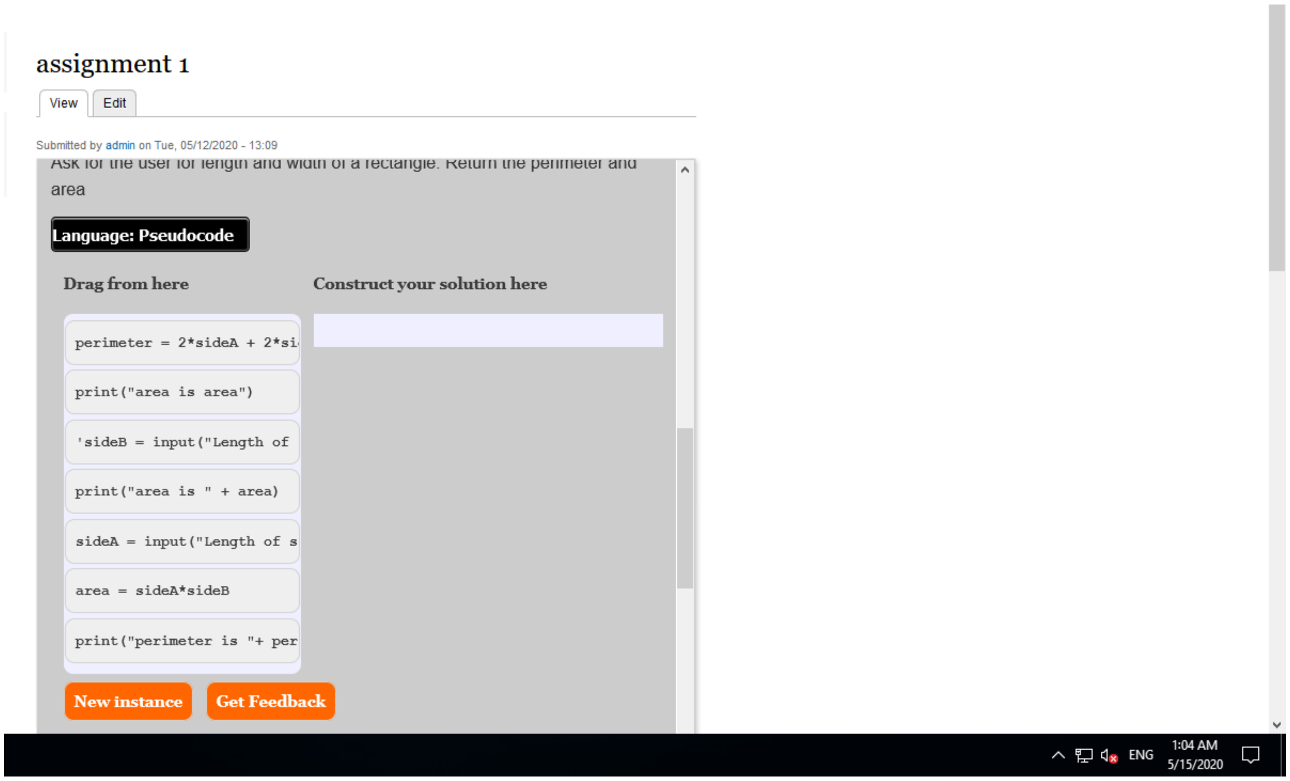
Quiz - 3

Due 8 Jun 2019 at 23:59 **Points** 5 **Questions** 5
Available 7 Jun 2019 at 9:10 - 8 Jun 2019 at 23:59 1 day **Time limit** None
Allowed attempts Unlimited

Attempt history

	Attempt	Time	Score
KEPT	Attempt 2	5 minutes	2.3 out of 5
LATEST	Attempt 2	5 minutes	2.3 out of 5
	Attempt 1	1,579 minutes	1.8 out of 5

Figure b: The assignment attempted by student will show the attempt history including the time taken and the score for every attempt.



assignment 1

View Edit

Submitted by admin on Tue, 05/12/2020 - 13:09

Ask for the user for length and width of a rectangle. Return the perimeter and area

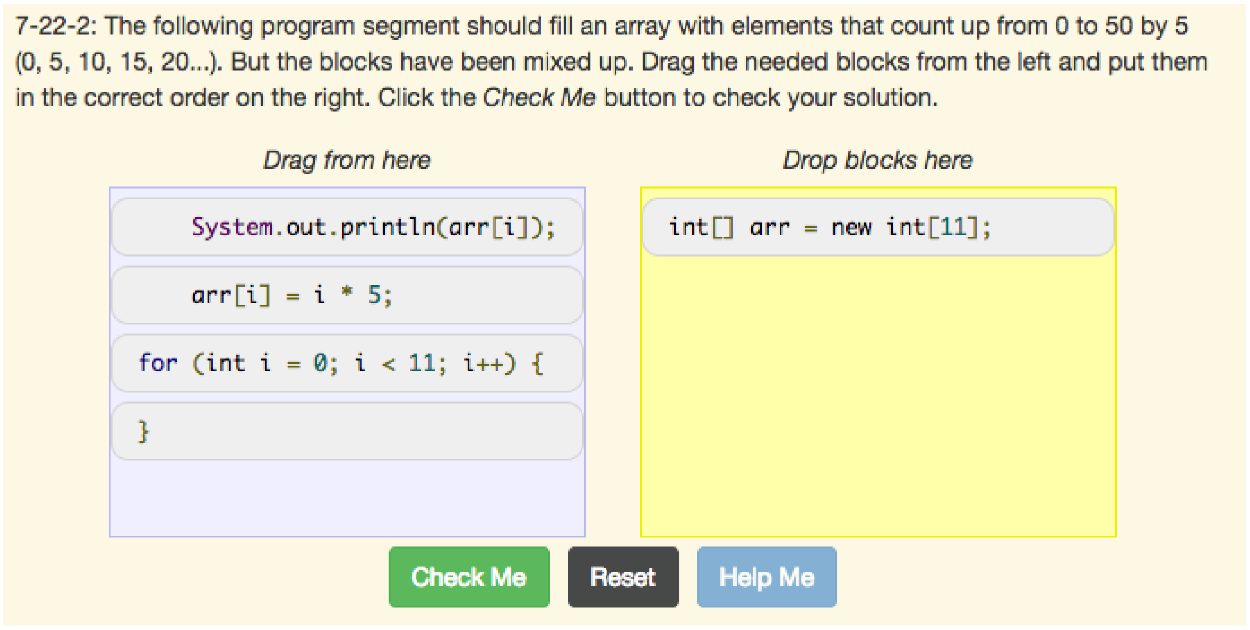
Language: Pseudocode

Drag from here

Construct your solution here

```
perimeter = 2*sideA + 2*si
print("area is area")
'sideB = input("Length of
print("area is " + area)
sideA = input("Length of s
area = sideA*sideB
print("perimeter is "+ per
```

Figure c: The parsons puzzle test for the student where the student will attempt the quiz.



7-22-2: The following program segment should fill an array with elements that count up from 0 to 50 by 5 (0, 5, 10, 15, 20...). But the blocks have been mixed up. Drag the needed blocks from the left and put them in the correct order on the right. Click the *Check Me* button to check your solution.

Drag from here

```
System.out.println(arr[i]);
arr[i] = i * 5;
for (int i = 0; i < 11; i++) {
}
```

Drop blocks here

```
int[] arr = new int[11];
```

Figure d: Student attempting the problem and creating the solution.

Tags of problem

Difficulty of the problem depends upon the following:

- Length of the code lines

- Increasing number of distractors
- Unpaired distractors are harder
- Hybrid of reorder/indentation/fill in the blank makes it harder
- Specific topics which are harder to understand - complex topics like:
    - algorithm – tree, calculate the combination, nested for-loop, recursive
    - sorting algorithm
    - data structure
    - the order of functions
    - the syntax of using pointer
    - order to build TCP connection(packet)
    - object – java (ask students to write class based on the needs of realizing some functions, interface) – fill in the blank
- Hybrid of multiple topics (making it bigger and more complex)

Tag Difficulty calculation: tag from the instructor + students' time spent/movement (actions taken to attempt)/try times/pass ratio data (after students submit the answer).

Tag difficult calculation considers the difficulty of the test/problem from the students as well. For example, student might create an easy difficulty test/problem but students are finding it hard. Hence, the feedback from students as well as the data while students attempt that test/problem is also considered and based on this calculation the difficulty for test/problem can be updated by the instructors.

Data analysis:

1. Generating data about the topics which are hard to comprehend: After collecting the data of all students from multiple subjects, models can be built to analyse the data. Information can be generated about the topic which are hard to comprehend. After receiving this information, the instructor can adjust the teaching plan or give some extra support on these topics.
2. Monitoring student progression: Data can also be used to monitor student's progression as to how students are performing and how the puzzles are helping them to develop their understanding as well as making it easier to comprehend the complex topics.
3. Analysing the learning pattern followed by each student. Instructors can divide the students into different groups based on their different cognitive processing strategies, regulation strategies and learning motivations. And then suggest them with different study plans and give them adaptive supports. For example, give more background or theoretical explanations to a meaning-directed learner. Arrange more practical sessions to application-directed learners.
4.

Option to extend it as "Dynamically adaptive parsons puzzle"

Create a model that checks the difficulty of the problem and publishes the problem depending on the tag as well as the time taken by the student to solve the last problem, the attempts taken by the student to correctly pass the last problem and the grade student gets after attempting the last problem. Basically, it chooses problem/test from the problem/test pool according the performance of the student in the last problem as well as the difficulty of the problem/test remaining in the pool.