

# BTech451 Semester Presentation

Dian LIN

# Code Runner Extension

**Introduction:** Push up barriers(code implementations) into Code Runner against cheating.

**Motivation:** The Functionality of cheating detection is not covered yet by Code Runner. Which may cause:

1. Unfair to hard work students
2. Lost faith in Code Runner
3. Students try to cheating in Code Runner

# Application/Tool(Preparation)

- **SandBox:** used to run the series of test cases for input source code under limited time which prevents infinite loops or deadlocks from blocking the system. And effectively prevent malicious codes.(guard system).
- **MySQL:** open-source relational database management system(RDBMS). Allowed users to create relationships between tables by primary keys and foreign keys.

# Ideas

- **Similarity checking:** the straightforward idea by checking submissions similarity. (research required)
- **Programming variation:** the answers of Code Runner questions could be obtained online or translated from different languages. (research required)
- **Functionality addition:** more functionality are possible to be implemented into Code Runner. (based on the research results)

# Research Process

## (Similarity Checking)

Length of Program	Similarity percentage
3 lines (shortest)	100%
3 - 67 lines	68% - 77%
67 lines(longest)	56.6%
Median Length: 30 lines	Median similarity:76.7%(relatively high)

Analysis: High similarity would occur due to short coding length, same question and example provided in lectures.

Conclusion: Similarity checking is not a feasible anti-cheating approach because of above reasons.

# Research Process

## (Programming variation)

- **Question types being selected:**

1. recursive question: summing numbers.
2. pre-define question: binary tree.
3. sorting question: bubble-sort, insertion-sort and merge-sort.

- **Analysis:**

1. Solutions of recursive and sorting questions are both easily found online, but pre-define question are not possible since Code Runner define Class for participants.
2. Programming languages translations is an unavoidable personal skill which is not able to be detected by Code Runner.

**Conclusion:** pre-defined questions become a preferred question creation way but can't prevent copy-paste cheating.

# Idea Implementation

- The promising idea to implement for Code Runner is to make sure every participant gets different questions.
- Idea: One Question has more than one variant, such as Boolean question for Odd or Even number.
- The more variants in one question, the more probability of cheating would be reduced.

# Proof of Concept

- Idea from last slides relates to question creation page but other functionality may remain what they used to be.
- We don't know yet if the idea is possible to be implemented and successfully working with MySQL Database, prototype should be made instead of directly moving to Code Runner.
- The GUI mock-up should be same as Code Runner but additional 'option' (variant) functionality needs to be added.
- All question details and test cases in prototype should be store into same database as Code Runner does.

# GUI Mock-up Implementation (Java)

Question Creation page:

Question Creation

Enter the Question Option:

Question Description

Sample Answer

Input test cases

test case 1

test case 2

Expected outputs

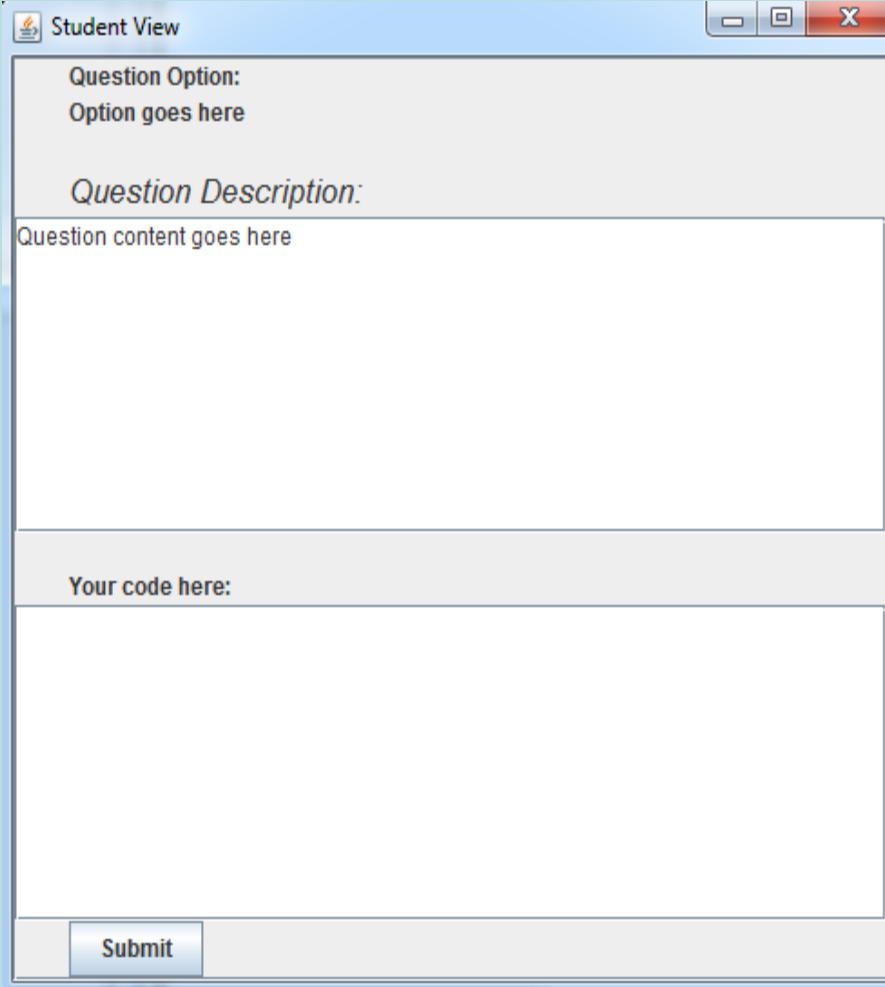
Expected output for test case 1

Administrator/Lectures variants view (after Odd and Even variants being added):

Question table below after first question being created						
A	B	C	D	E	F	G
Option	Question Descripti...	Sample Code	TestCase1	Expected output1	TestCase2	Expected output1
Odd	This is Odd test	public static boole...	System.out.println(...	false	System.out.println(...	true
Even	This is even test	public static boole...	System.out.println(...	true	System.out.println(...	false

# GUI Mock-up Implementation (Java)

- Student question view:

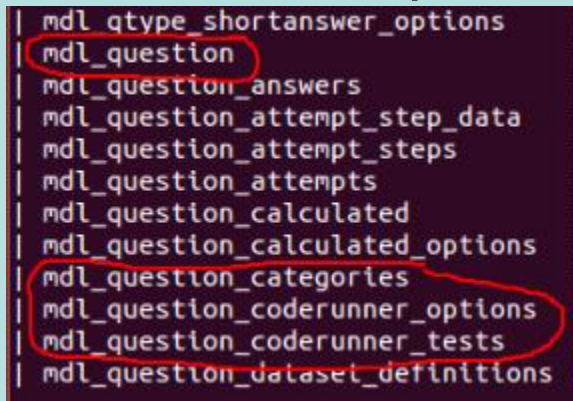


The image shows a Java Swing window titled "Student View" with standard window controls (minimize, maximize, close). The window contains a form with the following elements:

- A top section with a light gray background containing the text "Question Option:" and "Option goes here".
- A section below that with a light gray background containing the text "Question Description:".
- A large white text area below the description section, containing the text "Question content goes here".
- A section below that with a light gray background containing the text "Your code here:".
- A large white text area below the code section.
- A "Submit" button at the bottom left of the window.

# MySQL Database lookup

- All data displayed can be retrieved from Code Runner database, and what we need is to review what tables in the database relate to question creation. Records from following 4 circled tables are detected to be changed when created new questions.



A screenshot of a MySQL database interface showing a list of tables. The tables are listed in a vertical column, each preceded by a vertical bar. Four tables are circled in red: `mdl_question`, `mdl_question_calculated_options`, `mdl_question_categories`, and `mdl_question_coderunner_options`. The other tables in the list are `mdl_qtype_shortanswer_options`, `mdl_question_answers`, `mdl_question_attempt_step_data`, `mdl_question_attempt_steps`, `mdl_question_attempts`, `mdl_question_calculated`, `mdl_question_calculated_options`, `mdl_question_categories`, `mdl_question_coderunner_options`, `mdl_question_coderunner_tests`, and `mdl_question_dataset_definitions`.

mdl_qtype_shortanswer_options
mdl_question
mdl_question_answers
mdl_question_attempt_step_data
mdl_question_attempt_steps
mdl_question_attempts
mdl_question_calculated
mdl_question_calculated_options
mdl_question_categories
mdl_question_coderunner_options
mdl_question_coderunner_tests
mdl_question_dataset_definitions

Therefore, MySQL database is connected, and these 4 tables are used to store new records from my prototype.

# MySQL Database lookup

mdl\_question schema

Field	Type	Null	Key
id	bigint(10)	No	PRI
category	bigint(10)	No	MUL
name	varchar(255)	No	
questiontext	longtext	No	
qtype	varchar(20)	No	

mdl\_question\_categories schema

Field	Type	Null	Key
id	bigint(10)	No	PRI
name	varchar(255)	No	
contextid	bigint(10)	No	MUL

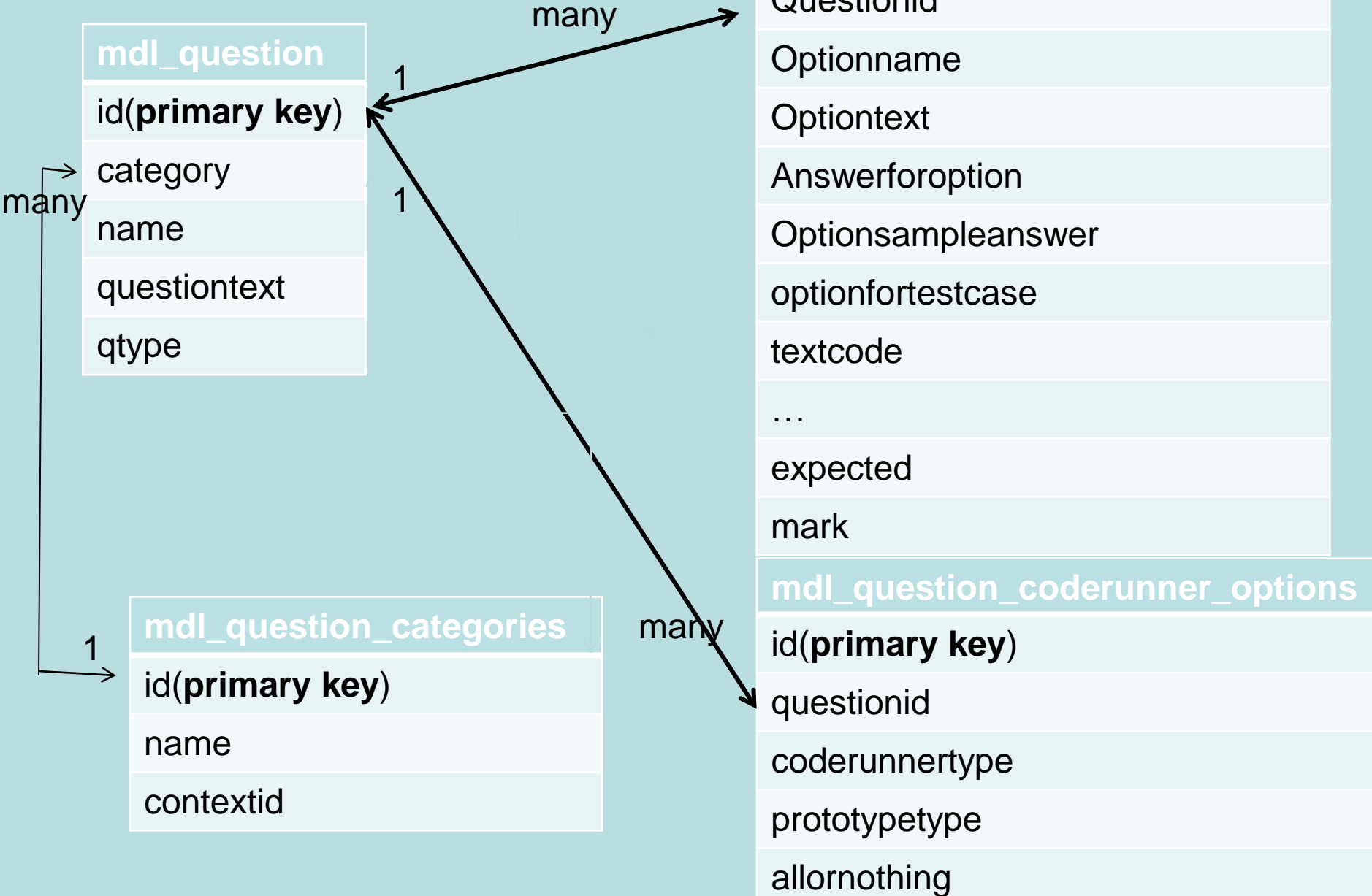
mdl\_question\_coderunner\_tests schema

Field	Type	Null	Key
id	bigint(10)	No	PRI
questionid	bigint(10)	No	MUL
testcode	longtext	YES	
stdin	longtext	YES	
expected	longtext	YES	
mark	decimal(8,3)	No	

mdl\_question\_coderunner\_options schema

Field	Type	Null	Key
id	begin(10)	No	PRI
questionid	bigint(10)	No	MUL
coderunnertype	varchar(255)	No	MUL
prototypetype	tinyint(1)	No	MUL
allornothing	tinyint(1)	No	

# Relationships of Tables



# Prototype Testing

- After GUI Mock-up and functionality implemented, we need to test if it is working as Code Runner does.
- Question creation test(record insertion)
  - In GUI Mock-up, type new question details, test case and expected output
  - all data above should be inserted into MySQL database as new records
  - question details should be able to display in both lecture's view and student's view

# Prototype Testing

Question Creation

Question option

Enter the Question Option:

Odd

Question Description

This is a test for emulating coderunner in java

Sample Answer

```
public static String checkOdd(int number){
    String isOdd = "false";
    if(number%2==1){
        isOdd = "true";
    }
    return isOdd;
}
```

Input test cases

test case 1

System.out.println(checkOdd(3));

test case 2

Expected outputs

Expected output for test case 1

true

Expected output for test case 2

Save

Delete

# Prototype Testing

```
mysql> select id,name from mdl_question;
```

id	name
1	BUILT_IN_PROTOTYPE_c_function
2	BUILT_IN_PROTOTYPE_c_program
3	BUILT_IN_PROTOTYPE_java_class
4	BUILT_IN_PROTOTYPE_java_method
5	BUILT_IN_PROTOTYPE_java_program
6	BUILT_IN_PROTOTYPE_octave_function
7	BUILT_IN_PROTOTYPE_php
8	BUILT_IN_PROTOTYPE_python2
9	BUILT_IN_PROTOTYPE_python3
10	BUILT_IN_PROTOTYPE_python3_w_input
13	Check if input is odd or even
14	alert test case
15	something
16	asdf
17	asfdfas
18	asfdasdf
19	asfdfas
20	asdf
36	Odd

```
mysql> select id,name from mdl_question_categories;
```

id	name
1	Default for CS101
2	Default for Miscellaneous
3	Default for System
4	Default for Front page
5	CR_PROTOTYPES
6	quiz
7	Default for new quiz
8	Default for second quiz
9	Default for Quiz Test
10	Default for Quiz2
11	Default for alert testing
12	Default for okok
13	Default for okok
14	Default for asdfas
15	Default for soemthing
16	Default for sfasdfa
32	Default for CS101

17 rows in set (0.00 sec)

[illegible][illegible]

**new records added  
to related tables**

# Prototype Testing

## Lecture and student view of the new created question

Question Creation

Input test cases

test case 1

test case 2

Expected outputs

Expected output for test case 1

Expected output for test case 2

Save

Delete

Question Preview

Student view

Question Option:  
1

Question Description:  
Odd

Your code here:

```
public static String checkOdd(int number){  
    String isOdd = "false";  
    if(number%2==1){  
        isOdd = "true";  
    }  
    return isOdd;  
}
```

Submit

Question table below after first question being created

A	B	C	D	E	F	G	H
Option Number	Option	Question Description	Sample Code	TestCase1	Expected output1	TestCase2	Expected output1
1	Odd	This is a test for emulating c...	public static String checkOdd...	System.out.println(checkOdd...	true		

# Prototype Testing

With correct and incorrect answer, prototype is able to run students input as Java file and return the feedback as SandBox does in Code Runner.

```
dian@dian-virtual-machine: ~/Desktop
dian@dian-virtual-machine:~/Desktop$ javac FirstImp.java
dian@dian-virtual-machine:~/Desktop$ java FirstImp
successfully
System.out.println(checkOdd(3));
true
You have passed test case!
```

Correct answer input and click 'submit'

```
dian@dian-virtual-machine: ~/Desktop
dian@dian-virtual-machine:~/Desktop$ javac FirstImp.java
dian@dian-virtual-machine:~/Desktop$ java FirstImp
successfully
System.out.println(checkOdd(3));
true
You have passed test case!
System.out.println(checkOdd(3));
true
clear
You have passed test case!
System.out.println(checkOdd(3));
true
You failed test case!
```

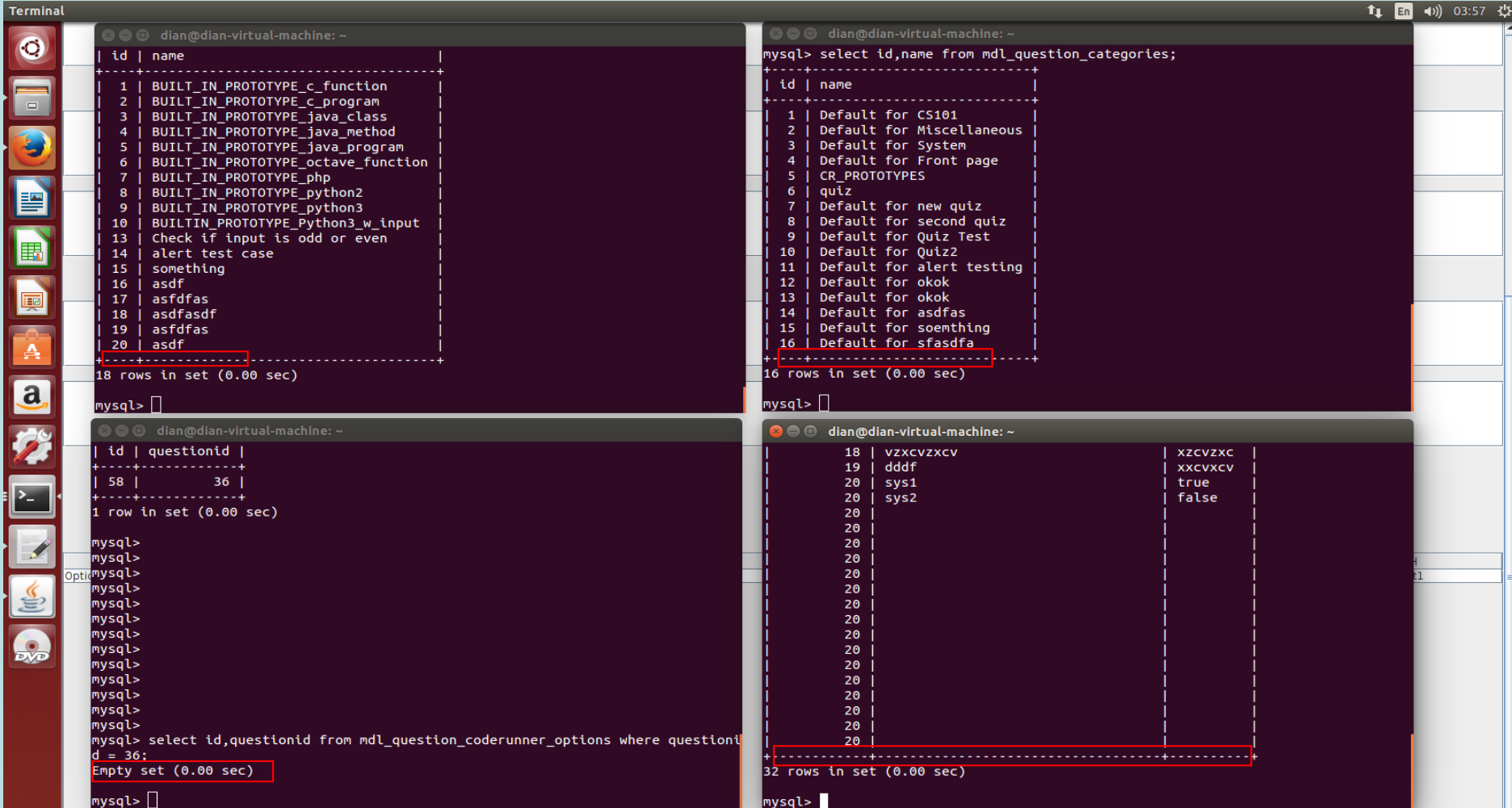
Incorrect answer input and click 'submit'

# Prototype Testing

- Deletion test:
  - records of the corresponding question will be both deleted from teacher's view and MySQL database
  - Students are no longer to view the deleted question again.

# Prototype Testing

The variants 'Odd' that we just created are now deleted from database.



The image displays four terminal windows from a virtual machine named 'dian@bian-virtual-machine'. The windows show the following database operations:

- Top Left Window:** A MySQL query `select id,name from mdl_question_categories;` returns a list of 20 categories. The last entry, 'asdfs', is highlighted with a red box. The output shows 18 rows in the set.
- Top Right Window:** A MySQL query `select id,name from mdl_question_categories;` returns a list of 16 categories. The last entry, 'Default for sfasdfa', is highlighted with a red box. The output shows 16 rows in the set.
- Bottom Left Window:** A MySQL query `select id,questionid from mdl_question_coderunner_options where questionid = 36;` returns an empty set, indicating that all related records have been deleted. The output shows 1 row in the set.
- Bottom Right Window:** A MySQL query `select id,questionid from mdl_question_coderunner_options where questionid = 36;` returns 32 rows, showing the state of the database after the deletion operation.

After delete button clicked for a specific record, all related record will be deleted from code runner database as well

# Further work

- So far, proof of concept has been successfully done by using Java programming in my prototype.
- Moving back to Code Runner and implement the concept by PHP programming skills.
- More feasible ideas against cheating need to be generated and implemented into Code Runner.
- Research based on particular ideas.

Thank you!

¿Questions?