1. Додавання задачі

operation – вказуэться операція

etalon – еталонний код виконання задачі, тобто код викладача. Має вирішувати задачу, бо з його результатом буде порівнюватись результат програми студента. Це поле може бути пустим!!!!

lang – вказується мова задачі, щоб знали чим компілити.

task – id задачі в базі

function\_name – як називатиметься функція студента/викладача

type – тип результату (того що вертає код викладача чи студента):

INT = 0,FLOAT = 1,BOOL = 2,STRING = 3, CHAR = 4

array\_type – чи є результат, який вертає код викладача чи студента, одновимірним масивом: 0 – ні, 1 – так.

size – розмір масиву, якшо результат масив. Якщо ні – інтерпретатор не бере значення цього поля до уваги.

unit\_test\_num – кількість юніт-тестів

checkable\_args\_indexes – фіча для порівняння будь-якої змінної, що фігурує в описі задачі, з будь-якою. Ці змінні беруться по індексам наступним чином:

0 – результат, який вернув код студента

1 – результат, який був вказаний як вірний у юніт-тесті

2 – результат викладача (еталонний

3 – значення першого вхідного параметра після виконання коду студента

4 – значення першого вхідного параметра, яким воно має стати після виконання коду студента (вказується в тесті)

5 - значення першого вхідного параметра після виконання коду викладача.

6 - значення другого вхідного параметра ….

Тут first порівнюється із second. Таких порівнянь в кожному тесті може бути багато, може не бути жодного. Але треба використовувати їх з розумом, бо ці порівняння впливають на результат тесту

Цих груп (виділив жовтим) має бути не більше, ніж є тестів. Тобто, якщо тестів 5, а груп порівняння 3, то відповідно вони будуть ваховуватись у перших 3 тестах Якщо ж їх буде більше, лишні просто не врахуються.

results – масив результатів. Використовується, тільки якщо еталонний код відсутній!!!!

Має мати розмір, який рівний кількості ю-тестів. Кожний його елемент має відповідати type, array\_type і size.

compare\_mark - Має мати розмір, який рівний кількості ю-тестів. Вказує, як в кожному тесті порівнюватиметься (який знак рівності) результат студента із результатом викладача

1. >=
2. >
3. ==
4. !=
5. <
6. <=

tests\_code - Має мати розмір, який рівний кількості ю-тестів. Навіщо воно – я не знаю, спроби забрати закінчились невдало. Просто має бути, пусті стрічки (“”,””,””,….)

args – описуються аргументи. В такому порядку як описані вони передаються в функцію

type, is\_array, size – аналогічно з описаним вище.

arg\_name – ім’я аргументу в коді

"pattern": {} – не знаю нашо, тоже хай буде.

value - Має мати розмір, який рівний кількості ю-тестів. Кожний його елемент має відповідати type, array\_type і size. Це початкове значення параметра в кожному тесті.

etalon\_value - Має мати розмір, який рівний кількості ю-тестів. Кожний його елемент має відповідати type, array\_type і size. Це значення параметру, яке очікується після виконання коду студента. Якшо нічого не очікуємо (не треба порівнювати), як значення берутьс пусті лапки

compare\_mark – аналогічно, знак порівняння параметру після виконання коду студента із очікуваним значенням.

{

"operation": "addtask",

"etalon": "return 3;",

"lang": "php",

"task": "110",

"function": {

"function\_name": "php\_110",

"type": 0,

"array\_type": 0,

"size": null,

"unit\_test\_num": 1,

"checkable\_args\_indexes": [

[

{

"first": 0,

"second": 1

},

{

"first": 2,

"second": 3

}

]

],

"results": [

"3"

],

"compare\_mark": [

2

],

"tests\_code": [

""

],

"args": [

{

"type": 0,

"is\_array": 0,

"size": null,

"arg\_name": "x",

"pattern": {},

"value": [

"3"

],

"etalon\_value": [

""

],

"compare\_mark": [

2

]

}

]

}

}

Інтерпретатор повертає:

{ "status" : "failed" } – така задача вже існує, і не була оновлена, так як нічого, в порівнянні з тим що є в базі, не змінилось.

{"id" : "114", "status" : "updated", "table" : "assignment" } – оновлено задачу 114.

{ "status" : "failed code compilation: error in line 1: ‘asdfasdfareturn’ was not declared in this scope\nerror in line 1: expected ‘;’ before ‘\_\_null’\n" } – якщо введено еталонний код, який не компілиться, верне помилку компіляції.

1. пуск коду студента на компіляцію

lang – вказується мова задачі, щоб знали чим компілити.

task – id задачі в базі

code – код студента

session і jobid – ідентифікують кожнний запуск студентом його коду на компіляцію. Тобто вони є якби парним ключом, і при кожномукиданні на компіляцію jobid має змінюватись.

{

"operation": "start",

"session" : "124 1qf ff22 ff4f 341",

"jobid" : 11212,

"code" : "

string arr=\"сука пизда\";

return arr;

",

"task": 110,

"lang": "c++"

}

Вертає:

{ "status" : "already exist" } – з таким йобідом і сесією запрос на компіляцію вже є

{ "status" : "Added to compile" } – поставилось в чергу на коспіляцію.

1. перевірка, чи код студента пройшов тести

"operation": "result",

"session" : "124 1qf ff22 ff4f 341",

"jobid" : 11212,

"task": 110,

"lang": "c++"

}

вертає:

{ "date" : "2016-04-27", "done" : true, "result" : " ", "status" : "done", "testResult" : [ true ], "warning" : "" }

тут

"done" : true – скомпілилось без помилок

"status" : "done" – пройшо всі тести

"testResult" : [ true ], - показує які тести пройшо, а які ні

{ "date" : "2016-05-04", "result" : "", "status" : "failed", "testResult" : [], "warning" : "error in line 2: expected initializer before ‘arr’\nerror in line 3: ‘arr’ was not declared in this scope\n" }

status" : "failed – не скомпілилось

warning – помилка компіляції