ОЧЕНЬ ВАЖНО!!!
Сообщение из БД "0 OK" должно содержать только английские буквы!!!

Общий вид процедуры/функции:

- [Имя функции]
Перефаваемые значения в процедуру/функцию: [параметр_1, параметр_2, ..., параметр_n]
Возвращаемые значения из процедуры/функции: [значение_1, значение_2, ..., значение_n]
Описание процедуры/функции

1) Процедуры для сервера

1.1) **Регистрация (запрос на регистрацию)**
[mail, password, login]
[code]
code = 0 OK - регистрация возможна
code = -1 ERROR - возникла ошибка
code = 1 WARNING - пользователь с таким логином уже существует в системе
code = 2 WARNING - пользователь с данной почтой и паролей уже существует в системе
Процедура должна проверить, допустима ли регистрация для данного пользователя или нет

1.2) **Регистрация (процедура регистрации)**
[mail, password, login]
[message]
message = 0 OK - регистрация прошла успешно
message = -1 ERROR - возникла ошибка
Процедура позволяет создать нового пользователя в таблице "User" и в связанных с ней таблице (таблице "Achievements")

1.3) **Авторизация**
[mail, password]
[message]
message = 0 OK - пользователь существует в системе
message = -1 ERROR - возникла ошибка
Процедура позволяет проверить, существует ли данный пользователь в таблице User

1.4) **Получение списка комнат**
[]
[список комнат из таблицы Room, в которых есть свободное место и у которых статус = "open"]
Процедура позволяет получить список комнат с помощью запроса SELECT

1.5) **Получение списка достижений для выбранного пользователя**
[mail, password]
[строка достижений для выбранного пользователя]
Процедура позволяет получить список достижений выбранного пользователя с помощью запроса SELECT

1.6) **Создание комнаты**
[owner_ID, max_count_of_players]
[room_ID]
Процедура позволяет создать запись в таблице "Room" и в связанных с ней таблицах, заполнить часть полей значениями по умолчанию и вернуть ID комнаты

2) Процедуры для комнаты.

2.1) **Подготовка к началу игры**
2.1.1) **Добавление игрока в комнату**
[mail, password, room_ID]
[count_of_players, max_count_of_players (из таблицы Room), user_ID (из таблицы User)]
Процедура позволяет создать запись в таблице "Player" и в связанных с ней таблицах. Позволяет получить данные из таблицы "Room" и значение поля "user_ID" из таблицы "User"

2.1.2) **Получение 3 * n случайных персонажей (без повторений)**
[3 * n]
[character_ID_11, character_ID_12, character_ID_13
character_ID_21, character_ID_22, character_ID_23
...
character_ID_n1, character_ID_n2, character_ID_n3]
Процедура позволяет получить 3 * n случайных без повторений значений поля "ID" из таблицы "Characters" и вернуть значение как при SELECT запросе,
где первый столбец - "character_ID_1", второй столбец - "character_ID_2", третий столбец - "character_ID_3"

2.1.3) **Получение n случайных ролей (НО ПО ПРАВИЛАМ ВЫДАЧИ РОЛЕЙ!!!)**
Правила выдачи ролей:
4 игрока: 1 шериф, 2 бандита, 1 ренегат
5 игроков: 1 шериф, 2 бандита, 1 ренегат, 1 помощник
6 игроков: 1 шериф, 3 бандита, 1 ренегат, 1 помощник
7 игроков: 1 шериф, 3 бандита, 1 ренегат, 2 помощника
[n]
[role_ID_1, role_ID_2, ..., role_ID_n]
Процедура позволяет получить n случайных значений поля "ID" из таблицы "Roles" по правилам выдачи ролей

2.1.4) **Добавление выбранного персонажа к игроку**
[player_ID, character_ID]
[code]
code = "0 OK" - записи успешно созданы
Процедура позволяет создать запись в таблице "Character" и связанных с ней таблицах ("Player" и "Characters"). Возвращает код в зависимости от успешности выполнения процедуры

2.1.5) **Добавление роли к игроку**
[player_ID, role_ID]
[code]
code = "0 OK" - записи успешно созданы
Процедура позволяет создать запись в таблице "Role" и связанных с ней таблицах ("Player" и "Roles"). Возвращает код в зависимости от успешности выполнения процедуры

2.1.6) **Выдача стартовых карт игроку**
[player_ID]
[card_ID_1, card_ID_2, ..., card_ID_n]
Процедура позволяет создать записи в таблице "Card", связать данные записи с соответствующей записью в таблице "Player" и вернуть ID значения добавленных карт
Рекомендации: сначала создать запрос на то, какое количество начальных карт должен получить игрок в начале игры (столько, сколько у его персонажа максимальное количество жизней), затем вызвать процедуру из пункта 3 на выдачу n количества карт игроку

3) Процедуры для реализации игрового процесса (для розыгрыша карт)

3.1) **Приём сообщение о начале/окончании хода игрока**
[player_ID, 'true/false']
[code]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет изменит значение поля "player_move" в таблице "Player" для заданного player_ID

3.2) **Проверка на наличие заданной карты у игрока**
[player_ID, card_ID]
[result]
result = "YES" - если карта имеется
result = "NO" - если карта отсутствует
Процедура позволяет проверить, существует ли связь между player_ID и card_ID

3.3) **Проверка на наличие заданной карты персонажа у игрока**
[player_ID, character_ID]
[result]
result = "YES" - если карта персонажа имеется
result = "NO" - если карта персонажа отсутствует
Процедура позволяет проверить, существует ли связь между player_ID и card_ID

3.4) **Проверка возможности выстрелить в игрока (учитывается расстояние между игроками и дальность стрельбы из оружия)**
[player_ID, target_ID]
[result]
result = "YES" - возможно выстрелить в игрока
result = "NO" - невозможно выстрелить в игрока
Процедура проверяет: ("firing_range" (таблица "Weapon") + "additional_attack_range" (таблица "Player")) (для player_ID) ==
("additional_defence_range" (таблица "Player") + "range" (таблица "Players_range")) (для target_ID)
Если истина, то result = "YES", иначе result = "NO"

3.5) **Выдача n карт игроку из колоды**
[player_ID, n]
[card_ID_1, card_ID_2, ..., card_ID_n]
Процедура позволяет изменить поля "card_location" и "player_ID" в таблице "Card" (ID карты выбирается по значению поля "count_of_cards" в таблице "Deck") и связать данные записи с соответствующей записью в таблице "Player" и вернуть ID значения добавленных карт. Также меняется значение поля "count_of_cards" в таблице "Deck"

3.6) **Выдача n карт игроку из сброса**
[player_ID, n]
[card_ID_1, card_ID_2, ..., card_ID_n]
Процедура позволяет изменить поля "card_location" и "player_ID" в таблице "Card" (ID карты выбирается по значению поля "count_of_cards" в таблице "Dropping") и связать данные записи с соответствующей записью в таблице "Player" и вернуть ID значения добавленных карт. Также меняется значение поля "count_of_cards" в таблице "Dropping"

3.7) **Переход карты игрока в сброс**
[player_ID, card_ID]
[code]
code = "0 OK" - карта успешно отправлена в сброс
Процедура позволяет для поля "player_ID" в таблице "Card" установить значение "NULL", изменить значения полей "card_location", "index_number" (данное поле меняется в зависимости от значения поля "count_of_cards" таблицы "Dropping"). Также меняется значение поля "count_of_cards" в таблице "Dropping"

3.8) **Выбранный игрок теряет n количество жизней**
[player_ID, n]

[code]
code = "0 OK" - процедура успешно выполнена
code = "10 OK" - процедура успешно выполнена, выбранный игрок погиб
Процедура позволяет уменьшить значение поля "lives" в таблице "Character" заданного игрока (если персонаж погиб, изменяется также значение поля "alive" в таблице "Player"
и значение поля "status" в таблице "Role")

3.9) **Выбранный игрок восстанавливает единицу здоровья**
[player_ID]
[code]
code = "0 OK" - процедура успешно выполнена
code = "1 WARNING" - у данного игрока максимальное количество жизней
Процедура позволяет увеличить значение поля "lives" в таблице "Character" заданного игрока

3.10) **Кража карты у игрока**
[plyaer_ID_from, player_ID_to, card_ID]
[code]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет изменить значение полей "player_ID" и "card_location" в таблице "Card"

3.11) **Получение списка карт, находящихся в руке у игрока**
[player_ID]
[card_ID, ..., card_ID]
Процедура позволяет получить список полей "ID" из таблицы "Cards", которые связаны с заданным полем из таблицы "Player"

3.12) **Установка нового оружия для игрока**
[player_ID, name, base_weapon, firing_range, endless_bang]
[code]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет изменить значения полей "name", "base_weapon", "firing_range", "endless_bang" в таблице "Weapon" заданного игрока

3.13) **Получение карты из колоды для проверки (проверенная карта уходит в сброс)**
[room_ID]
[card_ID, suit, rating]
Процедура позволяет получить значения полей "ID" из таблицы "Card" и значения полей "suit" и "rating" из таблицы "Cards" для заданной карты, исходя из значения поля "count_of_cards" в таблице "Deck". После получения значения поля "ID" из таблицы "Cards", изменяются следующие поля данной таблицы: "card_location" и "index_number". Также меняется значение поля "count_of_cards" в таблице "Deck" и значение поля "count_of_cards" в таблице "Dropping"

3.14) **Изменить дополнительную защиту игрока на n**
[player_ID, n]
[code]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет изменить значение поля "additional_defence_range" в таблице "Player" на заданное значение

3.15) **Изменить дополнительную дальность атаки игрока на n**
[player_ID, n]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет изменить значение поля "additional_attack_range" в таблице "Player" на заданное значение

3.16) **Проверка, есть ли у игрока на столе карта с аналогичным названием**
[player_ID, name]
[result]
result = "YES" - карта имеется у игрока
result = "NO" - карта отсутствует у игрока
Процедура позволяет проверить, существует ли записи в таблице "Card", где "player_ID" = player_ID (таблица "Card") и "card_location" = 4 (таблица "Card") и "name" = name (таблица "Cards")

3.17) **Получить n карт из колоды и добавить их на стол**
[room_ID, n]
[card_ID_1, ..., card_ID_n]
Процедура позволяет изменить поля "card_location" в таблице "Card" для первых n карт, лежащие сверху колоды (использовать значение поля "count_of_cards" в таблице "Deck") и вернуть ID значения добавленных карт. Также меняется значение поля "count_of_cards" в таблице "Deck"

3.18) **Передача карты игроку**
[player_ID, card_ID]
[code]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет изменить поля "card_location" и "player_ID" в таблице "Card" для выбранной карты

3.19) **Восстановление единицы жизни всем игрокам, если это возможно**
[room_ID]
[code]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет увеличить значение поля "lives" в таблице "Character" для всех игроков в заданной комнате, если значение поля "lives" < "max_count_of_lives"

3.20) **Получить n карт из колоды и добавить их в стадию выбора**
[room_ID, n]
[card_ID_1, ..., card_ID_n]

Процедура позволяет изменить поля "card_location" в таблице "Card" для первых n карт, лежащие сверху колоды (использовать значение поля "count_of_cards" в таблице "Deck") и вернуть ID значения добавленных карт. Также меняется значение поля "count_of_cards" в таблице "Deck"

3.21) **Возвращение карты в колоду**
[card_ID]
[code]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет изменить значение поля player_ID (на NULL), "card_location" и "index_number" в таблице "Card". Изменяется значение поля "count_of_cards" в таблице "Deck"

4) Процедуры для реализации игрового процесса (для изменения состояния)

4.1) **Победил ли игрок?**
[role_ID, role_ID]
[code]
code = "0 OK" - процедура успешно выполнена
Процедура позволяет изменить значение поля "win" таблицы "Player"