Конкурсное задание

Компетенция

Разработка решений с использованием блокчейн технологий (юниоры)

Конкурсное задание включает в себя следующие разделы:

1. Формы участия в конкурсе
2. Задание для конкурса
3. Модули задания и необходимое время
4. Критерии оценки
5. Необходимые приложения

Количество часов на выполнение задания:8ч.

1. ФОРМЫ УЧАСТИЯ В КОНКУРСЕ

Индивидуальный конкурс.

2. ЗАДАНИЕ ДЛЯ КОНКУРСА

Целью конкурсного задания является решение кейсов, связанных с построением или поиском неисправностей в блокчейн цепи.

3. МОДУЛИ ЗАДАНИЯ И НЕОБХОДИМОЕ ВРЕМЯ

Модули и время сведены в таблице 1

Таблица 1.

№ п/п	Наименование модуля	Рабочее время	Время на задание
1	Основные криптографические механизмы блокчейн системы	C1 10.00-12.00	2 часа
2	Работа с блокчейн цепью	C1 13.00-15.00	2 часа
3	Проверка целостности блокчейн цепи	C2 10.00-12.00	2 часа
4	Разработка и презентация решения	C2 13.00-15.00	2 часа

Ниже представлены задачи на понимание механизмов программирования узла блокчейн. Необходимые для работы файлы, указанные в заданиях приведены в архиве «**Задачи.rar**», расположенном на рабочем столе. Необходимые документы находятся во вложенных папках с названием соответствующем заданию.

Для представления решения необходимо создать на рабочем столе папку «МодульN». В папке «МодульN» для каждой задачи создается вложенная папка «Задача№», где N – это номер модуля, № - это номер задачи. В качестве решения прикладывается текстовый документ с ответом на задачу, а также проект, содержащий код решения.

Внимание!

При работе с файлами заданий, представленных в формате json, придерживайтесь следующих правил:

- используйте только двойные кавычки «"…"» для ключей и строковых значений;

- не используйте пробелы при формировании строки формата json:

  - {"value": 10} – **неверно;**

  - {"value":10} – **верно.**

Модуль 1. Основные криптографические механизмы блокчейн системы Блокчейн-сеть.

Задание №1

Найти хэш-значение транзакции, приведенной в формате json, используя алгоритм хэширования **sha384**. В качестве ответа укажите хэш значение в шестнадцатеричной форме. Транзакция содержится в файле **Модуль1/Task1/Task1-tx.json**.

```
{
    "from":"0xfc7e7d33639162659f8654129e34d199",
    "to":"0xa915c9750f35beffd6228136ae61f01e",
    "value":1631
}
```

Также приведены материалы для проверки алгоритма – в файлах **Task-sample-tx.json** и **Task1-sample-tx.txt** содержится пример транзакции и ее хэш соответственно.

InputExample:

```
{
    "from":"0x99a66b969f5b097dc7cd9e97dad0b259",
    "to":"0x9ecafc4ced49bd0bb357c885d05cb8a8",
    "value":1018
}
```

OutputExample:
4b447d3322f53b0dd6529b2201f2f29194c59ddabb51d030580f288b3900cb4c1409de1ed7dd81b33116f10442171995

Задание №2

Найти значение **nonce**, такое, чтобы хэш блока **начинался тремя нулями**. Блок данных представлен в файле «**Модуль 1/Task2/Task2_block.json**»

Хэш считается по алгоритму **SHA3-384**. Для того чтобы правильно посчитать хэш необходимо добавить в структуру блока поле '**nonce**', которое влияет на значение хэша. Значение этого поля имеет целочисленный тип. Блок представляет собой json-объект.

Итоговый вид блока:

```
{
   "index":387,

"pre_hash":"000426d85e5035e55d50f66b95978092dc667567b82a48f5123c2ec550135cc
f04b0e8f68983ec0a76d9b0146ab58eb7",
   "data":{
      "from":"Mary",
      "to":"Din",
      "value":559
   },
   "nonse":0,
   "hash":"_____"
}
```

Задание №3

Найти хэш-значение транзакции, приведенной в формате json, используя алгоритм хэширования **sha384**. В качестве ответа укажите хэш значение в шестнадцатеричной форме. Транзакция содержится в файле **Task3-tx.json**.

```
{

    "from":"0xd1afe79b06bce0bcecc1bd3872815072",

    "to":"0x00b6917d8246140a4871ebecbd118106",

    "value":398

}
```

Также приведены материалы для проверки алгоритма – в файлах **Task3-sample-tx.json** и **Task3-sample-tx.txt** содержится пример транзакции и ее хэш соответственно.

InputExample:

```
{
    "from":"0x4255fccff88514718e3e440006bd0fb4",
    "to":"0x6d6e91b6d654fccba616602a6fe7e32a",
    "value":1497
}
```

OutputExample:
37f8ae9bf16c92e8c28db3db3297aa2bf71184e31919d6407494d271a54185b227ab9373047bcb61ab497c36d1396eee

Задание №4

Известно исходное слово – «**WSJ2020**».
Преобразовать слово, используя **SHA3-256**. Затем полученный результат преобразовать в строку и сложить с **пятизначным** числом **x**. После этого сгенерировать хэш **SHA-256** и получить следующий хэш – «497e5015e87c682fbbfefb328a7912cbef5b27ab7c14ecaa32a4a12b9d864fc6» (Хэш для удобства приведен в файле «Модуль1/Task4/Task4-hash»). Определить значение хэша исходного слова по заданному алгоритму, а также минимальное число **x.**

Задание №5

Известно исходное слово – «**DistributedRegistry2020**». Преобразовать слово, используя **SHA-256**. Затем полученный результат преобразовать в строку и сложить с **пятизначным** числом x. После этого сгенерировать хэш **SHA3-384** и получить следующий хэш – «9cc985060c2b5f05ebda6d6b7341bf7b799025434ff1df384819a2660b8b6e9bb51aaec032859d1df5be6125662b5582». (Хэш для удобства приведен в файле «Модуль1/Task5/Task5-hash»). Определить значение хэша исходного слова по заданному алгоритму, а также минимальное число **x**

Модуль 2. Работа с блокчейн цепью

Задание №1

Вычислить цифровую подпись блока данных с помощью алгоритма RSA.

Каталог **Keys** содержит в себе закрытые ключи отправителей системы. Ключ создан при помощи алгоритма RSA, длина ключа 512 бит, алгоритм хэширования – **SHA-224**. Содержит поля d, e, n, p, q, используемые в работе криптографической системы.

Блок представлен в виде json-объекта и записан в файле **Модуль2/Task1/Task1-blok.json**.

Подпись должна быть представлена в шестнадцатеричном виде.

Задание №2

Для обеспечения целостности в блокчейн системах применяется метод построения дерева Меркле. Использование такого метода позволяет определить хэш-значение некоторого набора данных, например, транзакций в блоке. Дерево Меркле является бинарным, при его построении к каждому блоку данных применяется алгоритм хэширования, после чего, полученные значения записываются в листья дерева. Поднимаясь к корню, верви попарно объединяются, путем конкатенирования находящихся в них значений. Результатом вычислений является корень дерева – TopHash. Если на каком-то уровне дерева количество блоков данных нечетно, то крайний справа блок дублируется.

Участнику необходимо:

Определить количество уровней в дереве Меркля

Определить количество листьев в дереве Меркля

Построить дерево Меркля

Посчитать TopHash от текстовых строк, находящихся в файле «**Модуль2/Task2/Task2-tx.txt**». Ответом является хэш-значение. Используемый алгоритм хэширования – **SHA3-512**.

Для примера дается файл «**Модуль2/Task2/Task2-Emaple.txt**»

TopHash для этого примера является aa26d2f80ecae183b9226691c70e9963594f78607db1b78adb12e306d3ed7cda1d485960563a069e03185cfd7e94b3f42bd2f71b79aa746eaed81d607e704429 (содержится в файле «**Модуль2/Task2/Task2-Example-hash.txt**»)

Модуль 3. Проверка целостности блокчейн цепи

Задание №1

В некоторой блокчейн системе блоки данных хранятся в текстовых файлах в json-формате и имеют структуру вида:


```
{
    "index":1,
    "pre_hash":"000c87f4714b31eb8124d028fb1ce175c1e83f16f40e0972ecd22169fa3059e5",
    "data":[
        {
            "from":"Yana",
            "to":"Xaver",
            "value":1580
        },
        {
            "from":"Yana",
            "to":"Xaver",
            "value":248
        },
        {
            "from":"Yana",
            "to":"Xaver",
            "value":714
        },
        {
            "from":"Yana",
            "to":"Xaver",
            "value":647
        }
    ],
    "datahash":"f67923ad0b67f0cd772b02d427574bc5eb663d6517efa474dafbb53c1dfc9fe3",
    "creator":"Xaver",
    "nonce":500,
    "hash":"00019217781ddf3ff27ced1e99b818735fa564ca8753b59a0b7474d875e79aa0",
    "sign":"71b3d16775930547839b988b444e775e1ddfea48d4f361a1296c0268dbe75d9eb3378dcca1b21a1ec23fb6c5f6d084c7cb45a4c12c78eb9339117285a7766ed5"
}
```


Такая структура представлена следующими полями:

index – номер блока в цепочке;

pre_hash – хэш предыдущего блока;

data – набор транзакций в json-формате вида

from – идентификатор пользователя, совершающего перевод средств;

to – идентификатор пользователя, на чей счет переводятся средства;

value – объем средств;

datahash – хэш-значение полученное путем вычисления корня дерева Меркле от набора транзакций в блоке с помощью алгоритма **SHA-256**;

creator - идентификатор пользователя, сгенерировавшего блок;

nonce – поле, необходимое для вычисления хэша согласно протоколу PoW системы;

hash – хэш-значение от набора вышеперечисленных данных, представленных в формате json, начинается с3 нулей;

sign – цифровая подпись набора всех вышеперечисленных данных, учитывая хэш блока, представленных в формате json. Для подписи используется алгоритм RSA с алгоритмом хэштрования SHA-224

Архив **Keys** содержит файлы, в каждом из которых пять строк - ключевая информация каждого из пяти пользователей системы – поля **d, e, n, p, q** построчно, используемые в алгоритме RSA.

Архив **BlockChain** содержит несколько последовательных элементов цепочки (блоков). Некоторые файлы цепочки были преднамеренно повреждены.

Участникам необходимо:

Проверить элементы цепочки и обнаружить все поврежденные блоки, найти поврежденное поле и предоставить его правильное значение.

Формат ответа:

[номер блока] [поврежденное поле] [правильное значение]

Пример ответа:

5 datahash b0b5472863508f23c6fdbf8322209f1b00193b99e550f82fb5a1f6b71736b400

Такой ответ значит, что в пятом блоке значение datahash было повреждено, и правильным значением поля является b0b5472863508f23c6fdbf8322209f1b00193b99e550f82fb5a1f6b71736b400

Задание №2

Банковская система использует распределенный реестр для хранения транзакций, совершаемых пользователями.

Каждому пользователю доступны следующие операции со средствами:

-положить деньги на свой счёт [input];

-перевести деньги любому пользователю системы [send] (соответственно, получить деньги от другого пользователя [receive]);

-вывести деньги со счета в виде наличных [cash].

Так же любой пользователь может увидеть свой баланс [balance] и список совершённых транзакций[tx_list].

Список транзакций приведен в файле **Task2-txlist.txt**.

Необходимо посчитать во сколько раз расходы пользователя, адрес которого указан в файле **Task2-adr.txt**, в первой половине года отличаются от расходов во второй половине.

Результат округлить до двух знаков после запятой.

Модуль 4. Разработка и презентация решения

Перед разработчиком стоит задача разработать архитектуру блокчейн-решения для организации работы агентства купли-продажи недвижимости.

Необходимо разработать архитектуру решения. Указать основные функциональные модули системы и их взаимосвязь. Отразить основные механизмы формирования блокчейн-цепи (транзакции, блоки, генезисблок, работа с форками и т.д.)

Презентовать разработанную архитектуру решения.

4. КРИТЕРИИ ОЦЕНКИ

В данном разделе определены критерии оценки и количество начисляемых баллов (субъективные и объективные) таблица 2. Общее количество баллов задания/модуля по всем критериям оценки составляет 100.

Таблица 2.

Раздел	Критерий	Оценки		
		Судейская	Объективная	Итого
A	Основные криптографические механизмы блокчейн системы	0	22	22
B	Работа с блокчейн цепью	0	28	28
C	Проверка целостности блокчейн цепи	0	25	25
D	Разработка и презентация решения	4	11	15
Итого =		4	86	90

Субъективные оценки - Не применимо.

5. ПРИЛОЖЕНИЯ К ЗАДАНИЮ

1. *Приложение 1. Архив «Задачи.rar», содержащий файлы с конкурсными заданиями.*