**Муниципальное общеобразовательное бюджетное учреждение**

**гимназия № 1 имени Филатовой Риммы Алексеевны муниципального образования городской округ город-курорт Сочи Краснодарского края**

**ИНДИВИДУАЛЬНЫЙ ПРОЕКТ**

**на тему:**

**«Теоретическое создание невзламываемого, закрытого соединения посредством использования особенностей поляризационно запутанных фотонов и его симуляции.»**

НАПРАВЛЕНИЕ: Квантовые технологии, моделирование и симуляция квантовых систем

ПРЕДМЕТНЫЕ ОБЛАСТИ: Физика, информатика

|  |  |
| --- | --- |
| **Работа допущена к защите:**  **«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.**  **Работа защищена:**  **«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.**  **Количество баллов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Председатель комиссии:**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **(подпись) (ФИО)**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Члены комиссии:** | **Выполнил:**  **Горбенко Рудольф**  **Павлович,**  **ученик 9 «Б» класса**  **Руководитель:**  **МОБУ гимназии № 1**  **им. Филатовой Р.А.** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | Сочи, 2025 |

**ОГЛАВЛЕНИЕ**

**ВВЕДЕНИЕ……………………………………………………………………….2**

**ГЛАВА 1. ТЕОРЕТИЧЕСКОЕ СОЗДАНИЕ НЕВЗЛАМЫВАЕМОГО, ЗАКРЫТОГО СОЕДИНЕНИЯ ПОСРЕДСТВОМ ИСПОЛЬЗОВАНИЯ ОСОБЕННОСТЕЙ ПОЛЯРИЗАЦИОННО ЗАПУТАННЫХ ФОТОНОВ (UNBREAKABLE, CLOSED JUNCTION BY EXPLOITING FEATURES OF POLARIZATION-ENTANGLED PHOTONS. ДАЛЕЕ - UCJBEFOPEP)…………………....………………………………………………3  
1.1 КРАТКОЕ ОБЪЯСНЕНИЕ ИСПОЛЬЗУЕМЫХ ПРОЦЕССОВ ………3**

**1.2 ТЕОРЕТИЧЕСКОЕ СОЗДАНИЕ UCJBEFOPEP …………..……………3**

**ГЛАВА 2. СОЗДАНИЕ СИМУЛЯЦИИ UCJBEFOPEP ……….…………….6**

**2.1 НАПИСАНИЕ КОДА ……………………………………………………….6**

**2.2 АНАЛИЗ РЕЗУЛЬТАТОВ РАБОТЫ ПРОГРАММ …………….………13**

**ЗАКЛЮЧЕНИЕ……………………………………………………………...…16**

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ……………………………..17 ПРИЛОЖЕНИЕ………………………………………………………………...18**

**ВВЕДЕНИЕ**

**Актуальность выбранной темы:** Во времена полного отсутствия интернет-анонимности и возможности “прочитать” любое, в том числе проводное, соединение, создание чего-то, что выйдет за рамки этого правила более чем актуально. Правительственная, административная и пользовательская связь может быть в безопасности, благодаря моему устройству – UCJBEFOPEP. Это новый шаг в информационной анонимности.

**Проблема:** Возможность “прочитать” любое, в том числе проводное, соединение.

**Цель работы:** Придумать безопасный способ обмена информацией и доказать его работоспособность.

**Задачи работы:**

• Теоретически обосновать и объяснить принцип работы.

• Создать симуляцию устройства.

• Рассчитать примерную скорость передачи файлов.

**Объект исследования:** поляризационно запутанные фотоны.

**Предмет исследования:** Эффект квантовой запутанности, свойства поляризационно запутанных фотонов.

Г**ипотеза:** Возможно передавать информацию не давая возможности прослушки.

**Практическая значимость**: развитие подобной технологии поможет сделать информацию более анонимной.

**ГЛАВА 1. теоретическое создание невзламываемого, закрытого соединения посредством использования особенностей поляризационно запутанных фотонов (Unbreakable, closed junction by exploiting features of polarization-entangled photons. Далее - UCJBEFOPEP)**

**1.1 Краткое объяснение используемых процессов**

Квантовая запутанность — это явление, при котором квантовые состояния двух объектов становятся взаимосвязанными. Например, если при измерении спина первой частицы её спиральность положительная, то спиральность второй частицы всегда будет отрицательной. Проще говоря, это как два шарика, которые крутятся в противоположных направлениях. Я планирую использовать это свойство для шифрования информации.

Генерация второй гармоники (SHG) - это нелинейное взаимодействие волн, при котором два фотона с одинаковой частотой создают новый фотон с удвоенной энергией, что эквивалентно удвоенной частоте и уменьшенной длине волны в двое, сохраняя при этом когерентность возбуждения. Широко используется для удвоения частот лазеров. Это является частным случаем генерации суммарной частоты (для двух фотонов) и, более широко, генерации гармоник.

Спонтанное параметрическое рассеяние (SPDC) - оптический процесс, когда один фотон с высокой энергией (фотон накачки) преобразуется в пару фотонов с более низкой энергией (сигнальный фотон и холостой фотон). Позволяет генерировать запутанные пары фотонов.

**1.2 ТЕОРЕТИЧЕСКОЕ СОЗДАНИЕ UCJBEFOPEP**

Для генерации фотонов используем LED с длиной волны 200 нм.

Нам нужно попытаться получить фотон с как можно более высокой энергией, поэтому мы используем генерацию второй гармоники (SHG). Мы используем периодически поляризованный кристалл KTP (PPKTP) и преобразуем наш 200 нм луч в 100 нм с удвоенной энергией.

Далее мы будем использовать SPDC типа II. Для этого возьмём кристалл BBO (бета-борат бария) и направим на него луч светодиода. Большинство фотонов проходит через кристалл, но некоторые подвергаются спонтанному преобразованию вниз с поляризационной корреляцией типа II, образуя коррелированные пары фотонов. Эти пары имеют траектории, ограниченные сторонами двух конусов, оси которых симметрично расположены относительно луча накачки. Из-за сохранения импульса два фотона располагаются симметрично по сторонам конусов. Небольшая доля пар фотонов будет находиться на двух линиях, где пересекаются поверхности конусов, что приводит к запутанным поляризациям пар фотонов на этих линиях.

Помним, что полученная пара фотонов имеет вдвое меньшую частоту, и как следствие будет иметь длину волны в два раза больше, около 200 нм. Поэтому отсеиваем незапутанные фотоны фильтром, пропускающим длину волны от 150 нм.

Для засекания запутанных фотонов используем сверхпроводящий нанопроволочный однофотонный детектор (SNSPD или SSPD). Для доставки используем стандартный оптоволоконный кабель. Размещаем его между выходом фильтра и SSPD.

Объясню принцип дешифровки сигнала, который и будет лежать в основе моего проекта. У нас есть получатель, отправитель и источник, для начала рассмотрим устройство источника.

Берём светодиод, фокусируем луч линзами и направляем на PPKTP, полученный луч направляем на кристалл BBO. Ловим запутанные фотоны в оптоволокно, по которому один летит до получателя, а другой до отправителя.

Теперь рассмотрим устройство отправителя. Ловим фотон в оптоволокно и доставляем в кучу одинаковых линейных поляризаторов и ставим SNSPD или SSPD за ними. Теперь если направление начальной поляризации света и осью поляризатора совпало, то мы получим сигнал от SNSPD. Если сигнал есть, то считаем это за единицу, если нет, то ноль. Допустим наш запрос для отправки “1”, тогда если получаем из SNSPD 0, то отправляем по обычному проводному каналу связи 1 (то есть инвертировать), если же получаем 1, то отправляем 0 (не инвертировать). После отправки запроса таким образом отправляем восемь нулей для перевода в режим проверки. После перевода начинаем отправлять пароль, который система получателя знает заранее. И по кругу.

Теперь рассмотрим устройство получателя. Устройство идентично отправителю, за исключением того, что ось поляризатора отличается на 90 градусов. Если мы не в режиме проверки, то просто получаем 0 или 1 из SNSPD и инвертируем/не инвертируем, тем самым получая сообщение. Если мы в режиме проверки, то сравниваем пароль и уничтожаем систему при несовпадении. После проверки пароля от чужого отправителя говорим своему отправителю так же провести проверку.

Преимущество такой системы что, даже физически найдя кабели вы не сможете узнать сообщение не повреждая их. Для получения хоть одного сообщения будет необходимо создать такую же систему и обрезав кабеля от исходной системы подключить к своей, да и даже так вы не сможете получить больше одного сообщения, т.к. пока хоть один из получателей остаётся в системе, она будет уничтожена при несовпадении пароля, в то время как в обычной системе связи будет достаточно перерезать провода, идущие от отправителя.

Для аварийного уничтожения мы обкладываем наш источник взрывчаткой и подключаем к детонатору, который активируется если к нему перестанет поступать определённый сигнал.

**ГЛАВА 2.** **создание симуляции UCJBEFOPEP**

**2.1** **написание кода.**

Для начала создадим несложную симуляцию устройства.

Импорты библиотек:

ПРИЛОЖЕНИЕ6

Каждая библиотека нужна для определенной функциональности.

class PhotonSimulation:

def \_\_init\_\_(self, wavelength\_input=200e-9, crystal\_length=10e-3): Инициализация параметров симуляции, установка физических констант, создание структуры для хранения энергетической истории фотонов

ПРИЛОЖЕНИЕ30

def calculate\_photon\_energy(self, wavelength): Расчет энергии фотона по формуле E = hc/λ, использует постоянную Планка и скорость света, преобразует длину волны в энергию

ПРИЛОЖЕНИЕ31

def simulate\_interaction(self): Моделирование взаимодействия фотонов в генерации второй гармоники, расчет энергий входных и выходных фотонов, логирование энергетических параметров, вывод информации о генерации второй гармоники

ПРИЛОЖЕНИЕ32

def \_log\_photon\_energies(self, photon1\_energy, photon2\_energy): Сохранение энергетических данных входных фотонов, накопление статистики в истории энергий

ПРИЛОЖЕНИЕ7

def \_log\_output\_energy(self, output\_photon\_energy): Регистрация энергии выходного фотона, добавление в историю энергетических данных

ПРИЛОЖЕНИЕ8

Методы накапливают и выводят энергетические данные.

def \_wait\_for\_signal(self, filename, callback, interval=0.1): Реализует механизм межпроцессного взаимодействия через файл.

ПРИЛОЖЕНИЕ9

def plot\_energy\_dynamics(self): Создает график энергетической динамики фотонов.

ПРИЛОЖЕНИЕ10

class SPDCSimulation: Этот класс отвечает за моделирование спонтанного параметрического down-преобразования (SPDC).

def \_\_init\_\_(self, pump\_wavelength=100e-9, crystal\_type='BBO'): Инициализация параметров down-преобразования, установка физических констант, определение параметров кристалла

ПРИЛОЖЕНИЕ33

def simulate\_spdc\_type\_ii(self): Генерация спаренных фотонов, создание запутанного квантового состояния, случайный выбор поляризационных состояний, формирование антикоррелированной пары фотонов, преобразование в упрощённое бинарное представление вектора поляризации

Результат функции: Массив поляризационных векторов, запутанное квантовое состояние

ПРИЛОЖЕНИЕ34

def simulate\_down\_conversion(self): Расчет энергий входного, сигнального и холостого фотонов, применение формулы E = hc/λ, моделирование процесса параметрического down-преобразования

ПРИЛОЖЕНИЕ4

def process\_simulation(): Запуск полной симуляции квантовой коммуникации, ожидание сигналов от разных процессов, создание экземпляров PhotonSimulation и SPDCSimulation, выполнение их методов, запись и вывод результатов

ПРИЛОЖЕНИЕ35

def print\_simulation\_results(down\_conversion\_result, polarization\_result): Этот метод выводит на экран энергии входного, сигнального и холостого фотонов, а также информацию о спаренных фотонах.

ПРИЛОЖЕНИЕ5

def save\_polarization\_data(polarization\_result): Этот метод сохраняет данные о поляризации фотонов в JSON файлы для дальнейшего использования или анализа.

ПРИЛОЖЕНИЕ11

def main(): Запускает процесс симуляции в отдельном потоке, позволяя основному потоку ожидать его завершения.

ПРИЛОЖЕНИЕ36

Запуск программы:

if \_\_name\_\_ == "\_\_main\_\_":

while True:

main()

Этот блок запускает основную функцию в бесконечном цикле, что позволяет многократно выполнять симуляцию.

Теперь сделаем визуальную часть.

Импорты библиотек:

ПРИЛОЖЕНИЕ12

Класс SceneRenderer - основной класс для создания и управления 3D сценой:

def \_\_init\_\_(self): Создание 3D сцены, настройка matplotlib для визуализации, подготовка начальных условий

ПРИЛОЖЕНИЕ37

def setup\_initial\_conditions(self): Определение ключевых точек траектории, установка начальных параметров анимации

ПРИЛОЖЕНИЕ38

Статический метод rotate\_point() - поворот точки:

@staticmethod

def rotate\_point(point, angle\_deg):

ПРИЛОЖЕНИЕ1

Метод \_create\_cone\_3d() - создание 3D конуса:

@staticmethod

def \_create\_cone\_3d(height=5, radius=3, position=(0,0,0), color='cyan',

alpha=0.5, rotation\_angle=90):

ПРИЛОЖЕНИЕ13

def update(self, frame): Основной метод анимации, очистка кадра, отрисовка статических объектов, анимация точек

ПРИЛОЖЕНИЕ39

def \_entry\_stage(self, frame): Анимация входа фотона, движение точки от начальной до точки входа

ПРИЛОЖЕНИЕ40

def \_split\_stage(self, frame): Анимация разделения фотона, создание двух траекторий

ПРИЛОЖЕНИЕ41

def \_trajectory\_stage(self, frame): Анимация движения разделенных фотонов, отслеживание положения точек во времени

ПРИЛОЖЕНИЕ42

def run\_animation(self): Создание анимации с 170 кадрами

ПРИЛОЖЕНИЕ43

Методы создания куба (\_create\_cube):

@staticmethod

def \_create\_cube(position=(0, 0, 0), color='cyan'):

ПРИЛОЖЕНИЕ2

Метод создания цилиндра (\_create\_horizontal\_cylinder):

@staticmethod

def \_create\_horizontal\_cylinder(position=(1.37, 2, 0), length=1, radius=0.15, color='grey'):

ПРИЛОЖЕНИЕ3

def \_render\_static\_objects(self): Отрисовка статических геометрических примитивов, создание куба, конуса, цилиндра

ПРИЛОЖЕНИЕ44

def \_set\_scene\_limits(self):

ПРИЛОЖЕНИЕ14

@staticmethod

def \_save\_config(value): сохранение отчёта о начале процессов в JSON-файл

ПРИЛОЖЕНИЕ45

if \_\_name\_\_ == "\_\_main\_\_": Создание экземпляра класса SceneRenderer(), запуск анимации

ПРИЛОЖЕНИЕ46

def \_animate\_point(self, frame): Определение этапов анимации с границами кадров, выбор и выполнение текущего этапа

ПРИЛОЖЕНИЕ47

Теперь модифицируем код отправителя из теоретической части:

Импорт библиотек:

import json; import time; import datetime

Создадим функции:

read\_json\_file(filename): Читает и возвращает содержимое указанного JSON-файла. ПРИЛОЖЕНИЕ15

write\_json\_file(filename, data): Записывает данные в указанный JSON-файл. ПРИЛОЖЕНИЕ16

wait\_for\_signal(signal\_file): Ожидает, пока в файле не появится сигнал "1" (фотон долетел до приёмника), после чего сбрасывает его на "0". ПРИЛОЖЕНИЕ17

process\_input(message): Обрабатывает входное сообщение, сравнивая каждый символ с данными о поляризации фотона и записывает необходимо инвертировать или нет в config\_filename3.json. ПРИЛОЖЕНИЕ18

process\_sequence(length): Отправляет ‘0’ length раз на приёмник для перехода в режим пароля. ПРИЛОЖЕНИЕ19

process\_password(passw): Обрабатывает пароль, сравнивая каждый символ с данными о поляризации фотона и записывает необходимо инвертировать или нет в config\_filename3.json. ПРИЛОЖЕНИЕ20

Основная функция:

def main(): Выводится текущее время, считывается сообщение, и выполняются функции обработки входных данных и пароля

ПРИЛОЖЕНИЕ48

Теперь модифицируем код получателя из теоретической части:

Импорт библиотек:

import time; import json

Глобальные переменные:

global rezalt; global rez

read\_json(filename): Читает и возвращает содержимое указанного JSON-файла. ПРИЛОЖЕНИЕ21

write\_json(filename, data): Записывает данные в указанный JSON-файл. ПРИЛОЖЕНИЕ22

wait\_for\_signal(): Ожидает, пока в файле не появится сигнал "1" (информация о инвертировании появилась), после чего сбрасывает его на "0". ПРИЛОЖЕНИЕ23

def main(): Декодирование принятых поляризационных состояний, восстановление исходного сообщения, проверка целостности передачи

ПРИЛОЖЕНИЕ49

1. Состояние "s":
   * Ожидает сигнала с помощью wait\_for\_signal().
   * Читает данные о фотоне и инвертировании
   * Обрабатывает данные и добавляет их в список rez.
   * Если длина rez достигает 8, проверяет сумму элементов. Если сумма равна 0, переходит в состояние "password".
2. Состояние "password":
   * Проходит по каждому символу пароля.
   * Ожидает сигнала и считывает данные.
   * Проверяет, соответствует ли считанное значение ожидаемому. Если нет, переходит в состояние "sd".
   * Повторяется 4 раза
3. Состояние "sd":
   * Выводит сообщение 'sd' (system destroyed) и завершает выполнение программы.

Готовая симуляция - ПРИЛОЖЕНИЕ26, готовая визуализация - ПРИЛОЖЕНИЕ27, готовая модифицированная программа отправителя - ПРИЛОЖЕНИЕ28, готовая модифицированная программа приёмника - ПРИЛОЖЕНИЕ29.

**2.2** **анализ результатов работы программ.**

Проанализируем как работают программы

Когда вы вводите сообщение:

Каждый символ сообщения преобразуется в последовательность битов

Для каждого бита происходит квантовая генерация поляризационных состояний фотонов

Используется принцип квантовой запутанности

Механизм передачи:

Отправитель:

Ждет сигнала от симуляции через config\_filename4.json

Считывает сгенерированное поляризационное состояние фотона

Сравнивает его с битом сообщения

Записывает статус (совпадение/несовпадение) в config\_filename3.json

Получатель:

Принимает сгенерированное поляризационное состояние

Декодирует бит на основе полученного состояния

Накапливает биты в массиве rez

Особенности реализации:

Использование json-файлов как каналов коммуникации

Многопоточность для синхронизации процессов

Случайная генерация квантовых состояний

Проверка целостности передачи через контрольную последовательность

Стадии передачи:

Генерация фотонных пар

Кодирование битов в поляризационные состояния

Передача и декодирование

Проверка пароля для подтверждения связи

Проанализируем выводы программ:

Симуляция:

Начало взаимодействия фотонов в кристалле PPKTP

Фотон 1 - Энергия: 9.94e-19 Дж #энергия начальных фотонов

Фотон 2 - Энергия: 9.94e-19 Дж

…

Фотон 1 - Энергия: 9.94e-19 Дж

Фотон 2 - Энергия: 9.94e-19 Дж

Процесс генерации второй гармоники: #начало процесса

Входная длина волны: 200.00 нм #длина волны начальных фотонов

Выходная длина волны: 100.00 нм #длина волны получившихся фотонов

Выходной фотон - Энергия: 1.99e-18 Дж #энергия получившихся фотонов

SPDCSimulation #начало процесса

Энергия входного фотона: 1.9878e-18 Дж #энергия получившихся в генерации второй гармоники фотонов

Энергия сигнального фотона: 9.939e-19 Дж #энергия получившихся в SPDC фотонов

Энергия холостого фотона: 9.939e-19 Дж # энергия получившихся в SPDC фотонов

Пара 1:

Фотон 1: [1 0 0]

Фотон 2: [0 0 1] # упрощённое бинарное представление вектора поляризации [x y z], где x - горизонтальная поляризация, y - круговая поляризация, z - вертикальная поляризация.

Отправитель:

2024-12-24 21:00:05.612173 #начало работы

10011100 #наше сообщение

1 #значение x фотона доставшегося отправителю

…

0

2024-12-24 21:25:04.329485#конец работы

Приёмник:

[1] # первый бит сообщения

[1, '0'] # первые два бита сообщения

…

[1, '0', 0, '1', '1', 1, '0', 0] # все 8 битов сообщения

[[1, '0', 0, '1', '1', 1, '0', 0]] # сообщение

[0] # первый бит сообщения

….

[0, '0', 0, '0', 0, '0', 0, 0] # 8 нулей, значит переходим в режим пароля

Визуализация выглядит следующим образом ПРИЛОЖЕНИЕ50.

**ЗАКЛЮЧЕНИЕ**

В рамках данной работы был разработан и смоделирован инновационный метод квантовой коммуникации UCJBEFOPEP, основанный на использовании поляризационно запутанных фотонов.

Основные результаты исследования:

Теоретически обоснован принцип работы квантового канала связи с использованием эффекта квантовой запутанности:

Доказана возможность генерации запутанных фотонных пар

Разработан алгоритм кодирования и передачи информации через квантовый канал

Показана принципиальная невозможность перехвата информации без разрушения квантового состояния

Создана компьютерная симуляция устройства, включающая:

Моделирование генерации второй гармоники

Симуляцию спонтанного параметрического down-преобразования

Визуализацию процессов квантовой передачи

Экспериментально подтверждена гипотеза о возможности передачи информации с абсолютной защитой от несанкционированного доступа.

Практическая значимость работы заключается в:

Разработке принципиально нового подхода к защищенным коммуникациям

Создании теоретической базы для квантовых систем передачи информации

Демонстрации потенциала квантовых технологий в области информационной безопасности

Перспективы дальнейших исследований:

Совершенствование симуляции.

Создание физического прототипа устройства

Тестирование на реальных оптоволоконных линиях

Оптимизация параметров квантового канала связи

Проект UCJBEFOPEP представляет собой значительный шаг в развитии квантовых коммуникационных технологий и открывает новые возможности для обеспечения абсолютной информационной безопасности.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. <https://en.wikipedia.org/wiki/Quantum_entanglement>
2. <https://en.wikipedia.org/wiki/Second-harmonic_generation>
3. <https://en.wikipedia.org/wiki/Spontaneous_parametric_down-conversion>
4. <https://ru.wikipedia.org/wiki/Параметры_Стокса>
5. <https://ru.wikipedia.org/wiki/Линейная_интерполяция>
6. <https://en.wikipedia.org/wiki/Beer–Lambert_law>
7. <https://en.wikipedia.org/wiki/Polarizing_filter_(photography)>
8. <https://ru.wikipedia.org/wiki/Функция_Вигнера>
9. <https://en.wikipedia.org/wiki/Tensor>
10. <https://ru.wikipedia.org/wiki/Фазовый_синхронизм_в_нелинейной_оптике>
11. <https://en.wikipedia.org/wiki/Superconducting_nanowire_single-photon_detector>
12. <https://en.wikipedia.org/wiki/Polarizer>
13. <https://ru.wikipedia.org/wiki/Поляризатор>
14. <https://ru.wikipedia.org/wiki/Волоконно-оптический_кабель>
15. <https://ru.wikipedia.org/wiki/Фотоэлектронный_умножитель>
16. <https://en.wikipedia.org/wiki/Photodiode>

**ПРИЛОЖЕНИЕ**

Приложение1: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition1.py

Приложение2: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition2.py

Приложение3: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition3.py

Приложение4: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition4.py

Приложение5: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition5.py

Приложение6: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition6.py

Приложение7: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition7.py

Приложение8: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition8.py

Приложение9: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition9.py

Приложение10: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition10.py

Приложение11: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition11.py

Приложение12: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition12.py

Приложение13: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition13.py

Приложение14: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition14.py

Приложение15: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition15.py

Приложение16: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition16.py

Приложение17: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition17.py

Приложение18: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition18.py

Приложение19: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition19.py

Приложение20: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition20.py

Приложение21: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition21.py

Приложение22: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition22.py

Приложение23: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition23.py

Приложение24: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition24.py

Приложение25: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition25.py

Приложение26: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition26.py

Приложение27: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition27.py

Приложение28: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition28.py

Приложение29: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition29.py

Приложение30: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition30.py

Приложение31: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition31.py

Приложение32: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition32.py

Приложение33: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition33.py

Приложение34: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition34.py

Приложение35: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition35.py

Приложение36: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition36.py

Приложение37: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition37.py

Приложение38: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition38.py

Приложение39: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition39.py

Приложение40: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition40.py

Приложение41: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition41.py

Приложение42: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition42.py

Приложение43: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition43.py

Приложение44: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition44.py

Приложение45: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition45.py

Приложение46: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition46.py

Приложение47: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition47.py

Приложение48: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition48.py

Приложение49: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition49.py

Приложение50: https://github.com/RUD9/UCJBEFOPEP-simulation/blob/main/additions/addition50