**Задачи проекта:**

1. Оптимизировать время работы преподавателя, это включает в себя создание задач быстрее и более точные
2. Составить алгоритм, который помогает в создании n-количества типов задач
3. Понять, как можно измерить мощность множества задач, которое может быть сформировано в рамках этого количества задач
4. Составить классификацию типов задач, которые возможно генерировать в рамках проекта

**Типы задач, их подтипы и примеры:**

1. Синтаксис
2. Ошибки, подсвечиваемые редактором кода (убрать точку с запятой, сделать опечатку, удалить двоеточие (python)/скобки (c++), удалить/добавить табуляцию (python), использование недопустимого символа (-) или ключевого слова, недопустимая комбинация кавычек в строке, неверное название библиотечных функций (Print - python), дублирование переменных или функций с одинаковыми именами)
3. Добавление новых циклов
4. Замена типа переменных/класса
5. Изменение количества повторений в циклах
6. Замена названий между двумя и более переменными/функциями/классов в коде
7. Логика
8. Нарушение логики условий (замена знаков (<, <=, >, >=), неверное использование логических операторов (and, or, not – python)/(&&, ||, ! – c++))
9. Сравнение разных типов переменных (числа-строки, int – float (округление), списки-кортежи-словари, сравнение объектов классов, None – другие)
10. Ошибки в функциях (в неправильном порядке переданы аргументы в функцию, возврат не того аргумента)
11. Создание ошибок с указателями (c++), питон как доп. задача (неверное выделение/освобождение динамической памяти)
12. Область видимости (переменная внутри блока недоступна за его пределами, проблемы с глобальными переменными, затенение переменных)
13. Ошибка в обработке массивов или списков (неправильное обращение к элементам массивов или списка, выход за их границы, неправильное копирование элементов)
14. Замена одного цикла на другой (while на do while, for на while и т.д.)
15. Создание утечки памяти (с++)
16. Алгоритмы
17. Выполнение арифметических операций с элементами массива (поиск суммы/разности элементов с помощью цикла/рекурсии)
18. Требование выполнить конкретный алгоритм (поиск элементов (обход дерева в глубину/обход в ширину/рекурсивный обход), нахождение кратчайшего пути в графе (алгоритм Дейкстры, алгоритм Флойда-Уоршелла), нахождение МОД (алгоритм Прима, алгоритм Краскала), алгоритм Форда-Фалкерсона, строковые алгоритмы, перемножение/ сложение больших матриц, нахождение обратной и т.д.)
19. Требование: на входе сортировка. Замена одной сортировки на другую (пузырьком/быстрой сортировкой/слиянием/двоичная куча и др)
20. Оптимизация
21. Память. Добавление новых тяжеловесящих типов данных (замена всех int на double и вставка в формулы этих дополнительных данных); питон как доп. задача (замена обычных функций библиотеки numpy на функции, которые не выделяют лишнюю память)
22. Затирание кода
23. Затирание части строк
24. Затирание несколько строчек кода
25. Затирание реализаций некоторых функций, классов
26. Генерация теоретических вопросов. Требование: достаточно подробное описание кода
27. Принцип разработки
28. Применение принципов kiss, dry (реализация более простого решения, оптимизация дублирующегося кода)
29. Обработка исключений
30. Требование: код с ошибкой (деление на ноль, неправильный тип аргумента для функции или операции, обращение по несуществующему индексу)
31. Замена использование finaly-блока на синтаксис try-except блоков
32. Переписать код на другой язык
33. Написать документацию к коду