Минимальные правила кодирования.

1. Только C++.
2. Наличие среды разработки с поддержкой пошаговой отладки.
3. Наличие комментариев, объясняющих нетривиальные моменты в решении.
4. Необходимо придерживаться одного стиля написания кода.
   Возможные варианты:
   Google: http://google.github.io/styleguide/cppguide.html, автоматическая проверка: http://cpplint.appspot.com/
   ABBYY: https://drive.google.com/open?id=0B6oYnrxwl0UcLTRDVHNxVGxJeUU
5. Каждая сдаваемая задача предварительно должна быть сдана в eJudge.
6. В шапке сдаваемого cpp должно находиться закомментированное условие задачи.
7. Для проверки входных данных и работы встроенных функций рекомендуется использовать assert(...).
8. Массивы неконстантного размера, а также большого константного размера необходимо выделять в динамической памяти.
9. На каждый вызов аллокации памяти должен быть вызов освобождение памяти, new -> delete, malloc -> free (годится только для переменных базовых типов). Начиная со второго модуля штраф “-1” за любую утечку памяти в сдаваемом и показываемом коде.
10. Вызов delete должен быть в той же логической области видимости что и вызов new (либо в той же функции, либо в том же классе).
11. Код решения должен быть грамотно разбит на функции: в main должен находиться ввод данных, вызов функции, которая решает задачу, вывод результата. Ввод/вывод производится только в main, решение - только в отдельном наборе функций и/или классов.
12. Каждой структуре данных должен соответствовать класс с продуманным стандартным публичным интерфейсом.
13. Все переменные и функции должны иметь осмысленное имя. Однобуквенные имена разрешаются только для счётчиков цикла и для переменных, заданных в условии задачи.
14. Переменные должны объявляться по месту использования.
15. Все переменные базового типа должны при объявлении инициализироваться, классы и структуры должны иметь конструктор.
16. Глобальными переменными пользоваться нельзя.
17. Рекурсией произвольной глубины пользоваться нельзя. Ей можно пользоваться, только если вы уверены, что глубина не превысит 1000.
18. Для каждой задачи необходимо определить скорость работы и потребляемую память.
19. Реализованное решение должно быть максимально эффективным.
20. Любое дублирование чужого кода штрафуется “-5” и незачётом по задаче