Рекомендации к написанию кода и сдачи заданий

2020-2021

Правила игры

Этапы сдачи

Pending Review → Summon for Defence → OK

Сдача задач будет происходить посредством проверяющей системы **ejudge**. Как только ваш код пройдет тесты, статус посылки станет **PR** (Pending Review). Ревью делает семинарист. Если семинарист зачтёт посылку, статус поменяется на **SD** (Summon for Defence). Это означает, что вы должны лично защитить решение. Если защита успешна, то посылка получает статус **OK**, а баллы за задачу идут вам в копилку. Если же на каком-то из этапов проверяющему не понравился ваш код, то он пишет вам комментарий с требованиями что-то исправить и проставляет статус **Rejected**. За Rejected баллы не снимаются, но будьте внимательны при повторной отправке кода. Может случиться такое, что из-за невнимательности в код закрадётся какой-то баг и поломает его:с

По опыту прошлого года

Могу сказать, что границы перевода "еджаджовских" баллов в 10-тибалльную шкалу будет ближе к зачетной неделе. Поэтому на вопрос: "А скока задач надо на ***удосик***?", вы получите ответ, скорее всего, лишь в декабре.☹ Поэтому не стоит откладывать задачи в дальний ящик. Система штрафов довольно жесткая, что по просроченному решению, что по перепосылке. Она (система штрафов) испортила не один стул/кресло (выберите то, на чем вы сидите).
Также не стоит затягивать с дедлайнами каждого контеста. Лучше напрячься заранее и сделать всё самим, чем ночью судорожно искать человека, который уже сдал, и пытаться путем смены кодстайла заслать чужое решение. Это будет ощущаться при вашей защите решения.
Ещё не откладывайте сдачу задач ассистенту на конец года. В конце года не только у вас горят дедлайны поэтому больше чем обычно времени я скорее всего уделять не смогу.

А что будет на самой сдаче?

Что и как будет спрашивать Яковлев я не знаю. Ассистентам была дана настоятельная рекомендация досканально требовать объяснения каждой написанной строчки. Будьте также готовы ответить на пару-тройку простых вопросов из лекции/man/ридинга на github или продемострировать свои навыки, написав простой код на листочке. Очевидно, это будет очень просто человеку, который делает домашку сам.

Какой формат сдачи?

Периодически я буду создавать гугл таблицу с записью на сдачу. В ней каждый выбирает себе слот в указанный день в удобное для него время и пишет номера посылок, который он планирует защищать. Обычно, для защиты каждой задачи хватает нескольких минут. На сдачу в неделю будет примерно отводиться час-два, в зависимости от вашей и моей нагрузки. Имеется ввиду суммарное время затраченное проверяющим, а не сдающими. Надеюсь, вы все будете лапочками, будете всё знать, код будет прекрасным и вас никто более 30 минут задерживать не будет. Стандартные места для сдачи:

- в холле ЛК
- Диванчики ГК / свободные аудитории
- КДС 2ки
- Zoom конференция.

Если у кого-то есть острое желание провести сдачу, а ближайшие даты не устраивают(либо я продолбал создание таблички), то вы можете связаться со мной в тг либо написать в чатик. Можете не волноваться, для каждого найдётся время сдать задачки:)

Рекомендации по коду

Следующие требования нацелены на увеличение эффективности сдачи, а не на мои личные прихоти. Так мы с вами будем быстрее читать код.

C

1. Низкоуровневый кодстайл: ваш код должен соответствовать какому-либо разумному кодстайлу. В нем не должно быть смешения пробелов и табов, должно

быть единообразное обособление операторов пробелами, единообразное положение «{» и т. д.. Можно брать кодстайл из репозитория курса, можно
`sudo apt-get install indent`
`indent -bli4 -i8 -brs -brf -br main.c`.

2. Постарайтесь настроить свою IDE, чтобы весь код был единообразен. Неприятно видеть разный кодстайл внутри одной задачи..

3. Понятные названия для переменных - как и в курсе по алгоритмам, названия должны быть понятными (неоднобуквенные, исключение: счетчик цикла, индекс массива). Тем более, никакого транслита.

4. Разбивайте свой код на отдельные логические блоки с помощью функций.

5. Следуйте правилу чистых функций.

6. Разумное использование макросов. Можно: `#ifdef DEBUG`, `eprintf`, `assert`, ... Нельзя: `for_each`, `for_range`, макросы разворачивающиеся в целые блоки кода и т. п.

7. Следите за выделенной памятью. Будут утечки – буду требовать использование Valgrind. Также стоит освобождать память в той же области, где была выделена.

8. Пишите комментарии.

9. Используйте только нужные библиотеки. Мешанина из 40 строк одних импортов не приветствуется.

10. Не бойтесь использовать структуры (а когда используете, не забывайте про designated initializers). Они лишь помогают при восприятии кода.

Assembler

1. Делайте как можно больше комментариев. Возможно, описывайте, что хранится в регистрах после выполнения команды.

2. Разумно называйте метки.

3. Отступы также очень важны.

PS:

Те умнички, которые дочитали до конца, будут знать, что задачи с номерами 0 и 1 – обязательные, без них зачет получить нельзя.