1. Приведите описание 3 любых принципов тестирования как это может быть. (Например: есть парадокс пестицида, при регрессии у нас есть не изменяющиеся тест кейсы, хорошей практикой есть то, что их нужно пересматривать и актуализировать.)

А. Тестирование демонстрирует наличие дефектов, а не их отсутствие – то, что врач нашел сегодня у пациента проблемы с головой не дает гарантий, что через некоторое время у пациента не будет рака простаты. Дефекты не обязаны быть взаимосвязаны и обнаруживаться все и сразу в ходе только одного «тестирования», универсального «тестирования» не существует.

Б. Раннее тестирование сохраняет время и деньги – чем раньше начнется «тестирование», тем меньше будет изменений в ходе SDLC, тем меньше ресурсов задействовано, тем быстрее выпуск ПО и больше профит.

В. Заблуждение об отсутствии ошибок - нет здоровых людей, есть не дообследованные. У любого ПО могут быть баги, которые даже при самом «пристальном» и длительном тестировании можно не обнаружить пока бог не бросит кости и волей «случая» они обнаружатся. Тестирование может минимизировать появления риска, но не может утверждать, что этого риска не существует.

2. Нарисуйте пример клиент серверной архитектуры используя блок схему выдуманного сайта с возможными действиями пользователя и кодами ответов которые будут возвращаться в ответ на действия пользователя (постараться применить каждый тип кода ответа).

https://github.com/Randle923/Homework-27/upload

1. Банкомат может принимать карты своего банка, если вставлена карта другого будет ошибка и редирект на главную страницу. При вставлении карты банкомат запрашивает пин, если пин не верный будет ошибка, после 3 попыток банкомат забирает карту и выведет на главный экран. Если пин верный клиент может авторизоваться и может снять деньги или посмотреть баланс по карте. Из этих двух услуг можно выйти на главную.

Сделать схему состояний и переходов к данным требованиям.

https://github.com/Randle923/Homework-27/upload

1. Здание имеет 10 этажей и подвал (парковка). Как вы проверите лифты, которые могут идти только на нечетный и 2-й этаж? И какие техники можно применить.

Буду использовать метод «Черного ящика» с использованием техники «Эквивалентного разбиения». А также тестирование на основе опыта с «предположением об ошибках»

А вот «Анализ граничных значений» я не буду использовать т.к. класс «кнопок» должен состоять из **упорядоченных** числовых значений.

При «Эквивалентном разбиении» буду использовать следующие условия отбора

А. 0, 1, 2, 3, 5, 7, 9 – Необходимые (нечетные и «2») кнопки в лифте для позитивного сценария. При нажатии на них они должны отправлять лифт на выбранный этаж – это позитивный сценарий. При отправке на «другой» этаж или «не работе» лифта это негативный сценарий

Б. 4, 6, 8, 10 – Необходимые (четные кроме 2) кнопки в лифте для негативного сценария. При нажатии на них надо убедится что они **«не отправляют»** лифт на «выбранный» этаж, а не «другой» или лифт «не работает» при нажатии этих кнопок.

1. Привести конкретный пример S2 и S3 бага.

S2 – Критический баг (Critical) – Юзер пополняет мобильный телефон через Приват 24, деньги с расчетного счета Приват-банка уходят, но пополнение мобильного счета не происходит.

S3 – Значительный баг (Major) – Юзер делает попытку изменить пароль к своему аккаунту, но у него это не получается, ПО оставляет старый пароль без изменений.

1. У Вас есть сайт в котором найдено 20 ошибок, а протестировано 30 задач. Из 20 ошибок 3 нашли пользователи и 1 нашел разработчик, время потратили на тесты 30 часов.  Какие метрики можно сделать по этим данным и как они нам помогут?

Предисловие: пользователи это не тестировщики от компании, то, что они обнаружили 3 бага, не значит, что они это нашли в момент тестирования. Скорее всего они нашли их после официального тестирования. А вот разработчика с его 1 багом можно добавить в число «тестировщиков» от компании. Поэтому при «официальном» тестировании выявили не 20, а 17 шт багов. Это важно для понимания эффективности работы тимы тестировщиков.

1. Количество найденных багов «командой тестировщиков» за период времени:  
17 ошибок / 30 часов = 0,56 шт/час – понимание эффективности наших тестировщиков во времени.

2. Эффективность работы исключительно тестировщиков:

(20-3-1) багов/ 20 багов \*100= 80% ошибок нашли тестировщики (молодцы)

3. Сколько задач приходится на 1 баг:

1. 30 задач /20 ошибок = 1,5 задач/ошибку -этими данными можно «душить» разрабов во время утренней пятиминутки

4. Процент задач с багами:

1. шибок / 30 задач \*100 = 66,67% -этими данными можно «душить» разрабов во время утренней пятиминутки

5. Количество затраченного времени на тестирование 1 задачи

30 часов / 30 задач = 1 час/задачу – от этих данных можно задуматься о усовершенствовании процесса тестирования для уменьшения времени в дальнейшем.

6. Количество протестированных задач за 1 час работы

30 задач / 30 часов = 1 задача/час - – от этих данных можно задуматься о усовершенствовании процесса тестирования для уменьшения времени в дальнейшем.