|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| федеральное государственное автономное образовательное учреждение  высшего профессионального образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ФАКУЛЬТЕТ КИБЕРНЕТИКИ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ |
| КАФЕДРА КИБЕРНЕТИКИ (№ 22) |

«УтверждЕН

на заседании кафедры

«\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_ г.,

протокол №\_\_\_\_\_\_\_\_\_

зав.каф.22

\_\_\_\_\_\_\_\_\_\_\_\_\_/А.М. Загребаев/

**ФОНД ОЦЕНОЧНЫХ СРЕДСТВ**

по дисциплине

**«МЕТОДОЛОГИЯ ПРОГРАММНОЙ ИНЖЕНЕРИИ (ТЕХНОЛОГИЯ РАЗРАБОТКИ, ВЕРИФИКАЦИЯ И СЕРТИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ)»**

|  |  |
| --- | --- |
| Направление подготовки (специальность) | 09.04.04 Программная инженерия |
|  |  |
| Профиль подготовки (при его наличии) | Технологии разработки высококритичных кибернетических систем |
|  |  |
| Наименование образовательной программы (специализация) | Технологии разработки высококритичных кибернетических систем |
|  |  |
| Квалификация (степень) выпускника | магистр |
|  |  |
| Форма обучения | очная |

г. Москва, 2015 г.

**Паспорт**

**фонда оценочных средств**

**по дисциплине «Методология программной инженерии (технология разработки, верификация и сертификация программного обеспечения)»**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Семестр** | **Интерактив** | **Трудоемкость, кред.** | **Общий объем курса, час.** | **Лекции, час.** | **Практич. занятия, час.** | **Лаборат. работы, час.** | **СРС, час.** | **КСР, час.** | **Форма(ы) контроля, экз./зач./КР/КП** |
| 3 |  | 3 | 108 | 12 | 12 | 16 | 68 | 0 | З |
| ИТОГО | 0 | 3 | 108 | 12 | 12 | 16 | 68 | 0 |  |

Группа: К03-22М

**АННОТАЦИЯ**

Целью курса является ознакомление студентов с современными технологиями верификации программного обеспечения, приобретение ими навыков использования современных инструментальных средств для верификации программных систем. Курс знакомит студентов с методиками тестирования промышленных программных систем и охватывает вопросы построения тестового окружения, планирования системы тестов, анализа и улучшения покрытия тестируемой системы, интеграционного и нагрузочного тестирования, тестирования пользовательских интерфейсов. Также в программу курса входит рассмотрение процессов верификации проектной документации, а также процессов формальной инспекции кода и документации. Детально рассматриваются различные типы документации, сопровождающей процесс верификации, и процессы разработки, в которых эти документы участвуют. В завершение курса дается обзор методов разработки устойчивого программного обеспечения, позволяющих уменьшить продолжительность фазы тестирования без снижения его качества. Основным инструментальным средством, используемым при обучении, является Microsoft Visual Studio Team Edition for Software Testers.

По завершении курса студент овладевает всеми основными навыками, необходимыми для участия в верификации сложных промышленных систем.

В ходе изучения данной дисциплины предусматривается:

- проведение лекционных занятий в соответствии с утвержденной сеткой часов;

- самостоятельное дистанционное обучение с использованием материалов курса, размещенных на сайте Интернет-университета информационных технологий ИНТУИТ

- самостоятельное изучение литературы, поиск и анализ информации, размещаемой на сайтах, посвященных тестированию программного обеспечения, освоение теоретического материала, подготовка к семинарским занятиям;

- проведение лабораторных работ с использованием материалов курса, размешенных на сайте Интернет-университета информационных технологий ИНТУИТ

- сдача зачета по изучаемым темам.

В результате изучения дисциплины студент должен:

- обладать знаниями обо всех основных методах тестирования программного кода;

- иметь представление о комплексном тестировании программных систем;

- иметь представление о процессах, связанных с промышленной разработкой программных систем;

- знать методы создания надежных программных систем;

- иметь представление о стандартах качества в области разработки программного обеспечения;

**СТРУКТУРА ДИСЦИПЛИНЫ**

Общая трудоемкость дисциплины составляет 5 кр., 180 час.

Лекции: 12 час.

Практические занятия/семинары: 12 час.

Лабораторные работы: 16 час.

**КОМПЕТЕНЦИИ СТУДЕНТА, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ/ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ ОБРАЗОВАНИЯ И КОМПЕТЕНЦИИ СТУДЕНТА ПО ЗАВЕРШЕНИИ ОСВОЕНИЯ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ**

ОК-6 – способность проявлять инициативу, в том числе в ситуациях риска, брать на себя всю полноту ответственности

ОПК-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях

ПК-6 – понимание существующих подходов к верификации моделей программного обеспечения

ПК-11 – способность проектировать основные компоненты операционных систем

ПК-18 – владением навыками создания компонент операционных систем и систем реального времени

ПК-20 – владением навыками организации промышленного тестирования создаваемого программного обеспечения

**Структура и содержание учебной дисциплины**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ п.п** | **Наименование раздела учебной дисциплины** | **Недели** | **Лекции, час.** | **Практ. занятия/ семинары, час.** | **Лабораторные работы, час.** | **Обязат. текущий контроль (форма\*, неделя)** | **Аттестация раздела (форма\*, неделя)** | **Максимальный балл за раздел\*\*** | **Компетенции по разделам, проверяемые при текущем и рубежном контроле** | **Компетенции, проверяемые на зач. /экз.** |
|  | *3 семестр* |  |  |  |  |  |  |  |  |  |
| 1 | Методы тестирования и верификации программного обеспечения | 1-8 | 13 |  | 8 | БДЗ-8 | КИ, 8 | 30 | ОК-6, ОПК-5, ПК-6, ПК-11, ПК-18, ПК-20 | ОК-6, ОПК-5, ПК-6, ПК-11, ПК-18, ПК-20 |
| 2 | Процессы верификации и сертификации программного обеспечения | 9-18 | 14 |  | 10 | БДЗ-17 | КИ, 18 | 30 | ОК-6, ОПК-5, ПК-6, ПК-11, ПК-18, ПК-20 | ОК-6, ОПК-5, ПК-6, ПК-11, ПК-18, ПК-20 |
|  | *Итого за 3 семестр* |  | 27 | 0 | 16 |  |  | 60 |  |  |
|  | **Контрольные мероприятия после 3 семестра** |  |  |  |  |  | З | 40 |  | ОК-6, ОПК-5, ПК-6, ПК-11, ПК-18, ПК-20 |

**Контрольные вопросы по курсу «Верификация и сертификация программного обеспечения»**

Обучающемуся предлагается выбрать правильные ответы на поставленные вопросы. Структура вопросов по различным темам построена одинаковым образом: Пункт 1 – необходимо выбрать один правильный ответ.

Пункт 2 – нежелательно выбрать неправильный ответ.

Пункт 3 – желательно выбрать все правильные ответы

**Оценка ответов** производится по следующей схеме:

* Полный ответ оценивается в 5 баллов;
* Выбор неверного ответа в пункте 1 наказывается вычитанием одного балла;
* Выбор неправильного ответа в пункте 2 наказывается вычитанием двух баллов ИЛИ не выбор одного правильного ответа в пункте 2 наказывается вычитанием одного балла.;
* Не выбор любого правильного ответа пункта 3 наказывается вычитанием одного балла.

Таким образом, итоговая оценка может быть в диапазоне от 0 до 5 баллов.

**Раздел. Место верификации среди процессов разработки программного обеспечения»**

**Вопрос 1. Общее определение верификации**

Пункт 1. Верификация это

\*А. Процесс проверки соответствия поведения системы требованиям

Б. Процесс устранения ошибок в программном обеспечении

В. Процесс взаимодействия с пользователем, направленный на улучшение его понимания принципов работы программной системы

Г. Процесс уточнения требований по результатам обсуждения с пользователем

Пункт 2. В ходе верификации

\*А. Выявляются несоответствия поведения системы требованиям

Б. Устраняются ошибки и дефекты программной системы

\*В. Регистрируются выявленные дефекты и проблемы

\*Г. Составляются отчеты об устраненных проблемах

Пункт 3. Процесс верификации включает в себя

\*А. Управление выявлением ошибок

\*Б. Формальные инспекции

\*В. Тестирование программного кода

\*Г. Анализ недекларированных возможностей системы

**Вопрос 2. Типы жизненных циклов разработки ПО**

Пункт 1. Различия между каскадным и спиральным жизненным циклом заключаются в

А. Последовательности прохождения этапов

\*Б. Времени прохождения одной полной итерации цикла

\*В. Объеме реализуемой на каждом этапе функциональности

Г. В задействованных в разработке специалистах

Пункт 2. Вспомогательные процессы жизненного цикла

\*А. Направлены на создание инфраструктуры, необходимой для функционирования процесса разработки системы

Б. Могут отсутствовать в любом проекте без ущерба для получаемого результата

В. Характерны для каскадного жизненного цикла

\*Г. Включают в себя процесс гарантии качества и управления конфигурациями

Пункт 3. Жизненный цикл проекта по разработке программного обеспечения

А. Всегда определяется до начала разработки

Б. Не может меняться в ходе разработки

\*В. Имеет четко определенные результаты на каждом из этапов

\*Г. Регламентирует последовательность технологических операций в проекте

**Вопрос 3. Типы тестирования**

Пункт 1. Модульное тестирование предназначено для

А. Проверки функционирования одного замкнутого участка программного кода

\*Б. Проверки функционирования каждого независимого программного модуля

В. Для тестирования модуля в условиях отсутствия воздействия со стороны пользователя

\*Г. Для максимальной изоляции побочного влияния на функционирование модуля со стороны остальных частей системы

Пункт 2. Интеграционное тестирование предназначено для

А. Проверки корректной работы всех модулей после завершения их разработки

\*Б. Проверки корректности межмодульных интерфейсов

\*В. Постепенной проверки корректности совместной работы оттестированных модулей

\*Г. Уменьшения объемов системного тестирования

Пункт 3. Нагрузочное тестирование предназначено для

А. Проверки поведения системы на нештатных входных данных

\*Б. Проверки поведения системой при большом количестве обрабатываемых запросов

\*В. Проверки поведения системы при повышенной нагрузке на среду, в которой выполняется система

\*Г. Определения времени отклика системы на различных конфигурациях аппаратного обеспечения

**Вопрос 4. Вспомогательные процессы разработки ПО**

Пункт 1. Какие цели и задачи достигаются и решаются в процессе управления конфигурациями?

\*А. Управление изменениями данных

\*Б. Обеспечение целостности данных

В. Обеспечение целостности технологических процессов

\*Г. Обеспечение совместной работы коллектива разработчиков

Пункт 2. Какие цели и задачи достигаются и решаются в процессе гарантии качества?

\*А. Проверяется соответствие процесса разработки стандартам

Б. Дается гарантия того, что характеристики выпущенной продукции удовлетворяют некоторым критериям качества

\*В. Дается гарантия того, что процессы разработки дают возможность выпускать качественную продукцию

Г. Даеится гарантия отсутствия дефектов в разрабатываемой системе

Пункт 3. Какие виды процессов входят в жизненный цикл разработки ПО?

\*А. Процесс разработки системы

\*Б. Процесс верификации системы

\*В. Процесс управления проектом

\*Г. Обеспечивающие процессы

**Вопрос 5. Модель MSF**

Пункт 1. Из каких дисциплин состоит модель MSF?

\*А. Управление проектами

Б. Управление ресурсами

\*В. Управление рисками

\*Г. Управление подготовкой

Пункт 2. Из каких стадий состоит жизненный цикл в MSF?

\*А. Создание общей картины

\*Б. Планирование

\*В. Разработка

\*Г. Стабилизация

\*Д. Развертывание

Е. Сопровождение

Ж. Вывод из эксплуатации

Пункт 3. Какие компоненты составляют треугольник приоритетов в MSF?

\*А. Ресурсы

\*Б. Время

В. Потребности

\*Г. Возможности

**Вопрос 6. Роли в программирующем коллективе**

Пункт 1. Укажите основные свойства роли «Менеджер проекта»

\*А. Обеспечение реализации требований заказчика исполнителями проекта

\*Б. Взаимодействие с заказчиком

В. Разработка функциональных спецификаций

\*Г. Участие в приемо-сдаточных испытаниях

Пункт 2. Укажите основные свойства роли «Разработчик»

\*А. Участвует в разработке функциональных спецификаций

\*Б. Разрабатывает программный код

\*В. Консультирует тестировщиков в ходе тестирования

Г. Утверждает окончательный вариант тест-плана

Пункт 3. Укажите основные свойства роли «Специалист по сертификации»

\*А. Приводит документацию на систему в соответствие с требованиями сертифицирующего органа

Б. Принимает решение о типе получаемого сертификата

\*В. Обеспечивает коммуникацию между сертифицирующим органом и руководством проекта

\*Г. Определяет сертифицируемые свойства системы в рамках выбранного типа сертификации

**Вопрос 7. Роли в программирующем коллективе**

Пункт 1. Укажите основные свойства роли «Тестировщик»

А. Устраняет обнаруженные дефекты в системе

\*Б. Выявляет дефекты в системе

\*В. Участвует в разработке функциональных требований

\*Г. Создает отчеты о найденных дефектах

Пункт 2. Укажите основные свойства роли «Специалист по контролю качества»

А. Выявляет дефекты в системе

Б. Создает отчеты о найденных дефектах

\*В. Выявляет несоответствия процессов разработки установленным стандартам

\*Г. Дает рекомендации по улучшению процессов разработки

Пункт 3. Укажите основные свойства роли «Менеджер программы»

\*А. Планирует работы по проекту

\*Б. Участвует в разработке функциональных требований

В. Обеспечивает целостность проектных данных

Г. Обеспечивает продажи системы

**Вопрос 8. Тестирование, верификация и валидация**

Пункт 1. Дайте определение тестирования, как вида деятельности

\*А. Это процесс поиска и документирования дефектов программной реализации разрабатываемой системы

Б. Это процесс доказательства того, что программная реализация системы и требования на систему соответствуют друг другу и проектным стандартам

В. Это процесс доказательства того, что программная система соответствует ожиданиям пользователя или заказчика

Г. Это процесс поиска и исправления ошибок в проектной документации и программной реализации системы

Пункт 2. Дайте определение верификации, как вида деятельности

А. Это процесс поиска и документирования дефектов программной реализации разрабатываемой системы

\*Б. Это процесс доказательства того, что программная реализация системы и требования на систему соответствуют друг другу и проектным стандартам

В. Это процесс доказательства того, что программная система соответствует ожиданиям пользователя или заказчика

Г. Это процесс поиска и исправления ошибок в проектной документации и программной реализации системы

Пункт 3. Дайте определение верификации, как вида деятельности

А. Это процесс поиска и документирования дефектов программной реализации разрабатываемой системы

Б. Это процесс доказательства того, что программная реализация системы и требования на систему соответствуют друг другу и проектным стандартам

\*В. Это процесс доказательства того, что программная система соответствует ожиданиям пользователя или заказчика

Г. Это процесс поиска и исправления ошибок в проектной документации и программной реализации системы

**Раздел. «Методы тестирования.** **Тестовое окружение»**

**Вопрос 1. Процесс тестирования программного кода**

Пункт 1 Процесс тестирования программного кода включает в себя

\*А. выполнение исполняемого кода

\*Б. подготовку входных данных для тестирования

\*В. анализ результатов выполнения исполняемого кода

Г. устранение сбоев в работе программной системы

Пункт 2. Типичная процедура тестирования

\*А. основывается на подготовке и выполнении тестовых примеров под управлением тестового окружения

Б. основывается на анализе исходных кодов системы на наличие недекларированных возможностей

\*В. основывается на требованиях к тестируемой части программной системы

Г. основывается на результатах отладки, задокументированных программистами

Пункт 3. Типичные тесты программной системы выявляют следующие проблемы

\*А. несоответствие поведения системы требованиям

\*Б. неполноту требований

В. несоответствие требований потребностям пользователя

\*Г. неопределенность поведения системы

**Вопрос 2. Методы тестирования программного кода**

Пункт 1. Тестирование методом черного ящика подразумевает

\*А. полное сокрытие исходного текста от тестировщика

Б. доступность исходного текста в качестве справочного материала для тестировщика

\*В. определение входных и выходных значений для тестов только из требований

Г. полное отсутствие возможности проверить наличие недекларированного поведения системы

Пункт 2. Т естирование методом стеклянного ящика позволяет

\*А. проверить систему на наличие возможностей, не определенных требованиями

\*Б. проверить систему на наличие типичных ошибок, свойственных программистам

В. полностью автоматизировать процесс генерации тестовых примеров по коду

Г. заменить тестирование исходного кода на его модели

Пункт 3. Анализ программного кода

А. является полноценной заменой тестированию при сравнимой трудоемкости

\*Б. дополняет тестирование возможностью выявления трудноуловимых ошибок

В. может использоваться как основной источник информации для написания тестового окружения

Г. может использоваться как основной источник информации для генерации тестовых примеров

**Вопрос 3. Тестовое окружение**

Пункт 1. Тестовое окружение предназначено для

\*А. запуска и выполнения тестируемого модуля

\*Б. передачи входных данных и сбора выходных

\*В. отчуждения тестируемых модулей от системы

Г. локализации проблем в отдельных модулях системы

Пункт 2. Типичное тестовое окружение, состоящее из драйвера и заглушек

\*А. используется для отчуждения тестируемого модуля и имитации поведения системы

Б. используется для тестирования только одного модуля

\*В. может подавать на вход тестируемого модуля значения, которые никогда не будут переданы на его вход в реальной системе

Г. не меняется при любых изменениях тестируемого модуля

Пункт 3. Тестовое окружение

\*А. должно быть согласовано по интерфейсу функций и методов с тестируемым модулем

Б. должно моделировать всю функциональность системы, в состав которой входит тестируемый модуль

В. должно состоять только из драйвера и заглушек

Г. должно выполняться в режиме диалога с пользователем

**Вопрос 4. Тестовое окружение в объектно-ориентированных языках**

Пункт 1. Тестовое окружение в объектно-ориентированных языках

\*А. используется для тестирования дефектов, связанных с наследованием и полиморфизмом

\*Б. состоит из тестирующего класса и заглушек

В. всегда создает несколько объектов тестируемого класса

Г. использует защищенные данные тестируемого класса для своей корректной работы

Пункт 2. Выберите некорректные способы тестирования наследования в системе классов

А. замена части классов заглушками с теми же интерфейсами

\*Б. открытие всех защищенных и приватных данных классов

\*В. прямая модификация защищенных данных классов в обход интерфейса класса

Г. тест-драйвер работает с данными классов только через их интерфейсы

Пункт 3. Выберите правильные соответствия между элементами тестового окружения в структурных и объектно-ориентированных языках

\*А. драйвер = тестирующий класс

\*Б. заглушки функций = заглушки методов

В. заглушки функций = заглушки классов

\*Г. тестируемый модуль = тестируемый класс

**Вопрос 5. Тестовое окружение в событийно-управляемых системах**

Пункт 1. Тестовое окружение в событийно-управляемых системах

А. не содержит заглушек

Б. содержит только обработчики событий и не содержит других функций

\*В. часто моделируется при помощи конечных автоматов

Г. реагирует на все события таким образом, как это ожидает тестируемый модуль

Пункт 2.Тестовое окружение в событийно-управляемых системах

\*А. используется для тестирования взаимодействия между системами или модулями

Б. используется для тестирования совместного использования памяти

\*В. используется для генерации событий, воспринимаемых тестируемым модулем

\*Г. используется для генерации событий, которые не ожидаются тестируемым модулем

Пункт 3. Тестовое окружение в событийно-управляемых системах

\*А. передает и принимает события от тестируемого модуля

Б. обязательно компилируется в один исполняемый файл

В. используется только для тестирования отдельных модулей

Г. всегда работает согласно протоколу взаимодействия тестируемого модуля с внешним миром

**Вопрос 6. Определите корректный набор заглушек**

Пункт 1

Имеется следующий модуль:

#include “op\_float.h”

int sum(int a, int b)

{

}

int mult(int a, int b)

{

}

void main()

{

float a=5.5, b=6.5, c=0.0;

int d=1, e=2, f=0;

f=sum(d,e);

c=mult(a,b);

}

Определите функции, вместо которых должны быть написаны заглушки:

А. int sum(int a, int b)

Б. int mult(int a, int b)

В. float sum(float a, float b)

\*Г. float mult(float a, float b)

Пункт 2

Имеется следующий модуль:

#include “op.h”

int mult(int a, int b)

{

}

float mult(float a, float b)

{

}

void main()

{

float a=5.5, b=6.5, c=0.0;

int d=1, e=2, f=0;

f=sum(d,e);

c=mult(a,b);

}

Определите функции, вместо которых должны быть написаны заглушки:

\*А. int sum(int a, int b)

Б. int mult(int a, int b)

В. float sum(float a, float b)

Г. float mult(float a, float b)

Пункт 3

Имеется следующий модуль:

#include “op.h”

float sum(int a, int b)

{

}

int mult(float a, float b)

{

}

void main()

{

float a=5.5, b=6.5, c=0.0;

int d=1, e=2, f=0;

f=sum(d,e);

c=mult(a,b);

}

Определите функции, вместо которых должны быть написаны заглушки:

\*А. int sum(int a, int b)

Б. int mult(int a, int b)

В. float sum(float a, float b)

\*Г. float mult(float a, float b)

**Вопрос 7. Функции, выполняемые заглушками**

Пункт 1. Выберите верное утверждение

А. невозможно создание заглушек, не производящих никаких действий

\*Б. заглушки могут выводить информацию о значении внутренних переменных модуля

\*В. заглушки должны быть согласованы по интерфейсу с тестируемым модулем

Г. заглушки могут помещаться в исходный текст тестируемого модуля

Пункт 2.Выберите верное утверждение

\*А. заглушки могут использоваться без изменений разными драйверами

Б. заглушки всегда модифицируются при изменении внешних интерфейсов тестируемого модуля

В. скорость работы заглушек должна быть такой же, как и у заглушаемых функций

\*Г. заглушки могут имитировать работу аппаратного обеспечения

Пункт 3. Выберите верное утверждение

\*А. исходный текст заглушекможет помещаться в один файл с исходным текстом драйвера

\*Б. заглушки могут выводить в файл протокола значения, переданные им в качестве параметров

В. при несовпадении интерфейсов заглушаемых функций и заглушек тестовое окружение работает в режиме ограниченной функциональности

Г. диапазон значений, возвращаемых заглушками, должен совпадать с диапазоном значений, возвращаемых заглушаемымыми функциями

**Вопрос 8. Функции, выполняемые драйвером**

Пункт 1. Выберите корректные утверждения

А. драйвер вызывает заглушки в ходе тестирования

\*Б. драйвер вызывает функции тестируемого модуля в ходе тестирования

\*В. драйвер обеспечивает среду для работы тестируемого модуля

Г. драйвер это часть тестируемого модуля

Пункт 2. Выберите корректные утверждения

\*А. драйвер передает в тестируемыймодуль входные значения

\*Б. драйвер инициализирует среду выполнения тестов

В. драйвер собирает протокол вызова заглушек

Г. драйвер изменяет выходные данные, полученные от тестируемого модуля

Пункт 3. Выберите корректные утверждения

\*А. драйвер работает одновременно с тестируемым модулем

Б. драйвер может вызываться только один раз

В. драйвер отвечает за взаимодействе тестируемого модуля с операционной системой

Г. драйвер отвечает за взаимодействие тестируемого модуля с аппаратным обеспечением

**Раздел. «Тест-планы»**

**Вопрос 1. Типы тестовых примеров**

Пункт 1 Перечислите типы входных данных, обычно используемых для разработки тестовых примеров

\*А.допустимые данные

\*Б. недопустимые данные

\*В. граничные данные

Г. случайные данные

Пункт 2 Укажите минимально необходимый набор входных данных для тестирования арифметических функций

\*А. штатные данные, граничные данные, проверка на робастность

Б. штатные данные, случайные данные, недопустимые данные

В. граничные данные, проверка на робастность

Г. штатные данные, проверка на робастность, нештатное состояние среды выполнения

Пункт 3 Укажите количество и тип классов эквивалентности при тестировании функции сравнения двух строк. Классы необходимо выделять в зависимости от длины сравниваемых строк.

А. 3 класса: пустые строки, строки средней длины, строки максимальной длины

Б. 2 класса: строки средней длины, строки максимальной длины

В. 3 класса: пустые строки, строки максимальной длины, строки длиннее максимальной длины

\*Г. 4 класса: пустые строки, строки средней длины, строки максимальной длины, строки длиннее максимальной длины

**Вопрос 2. Тестирование операции сравнения**

Пункт 1. Допустимый интервал значений для чисел a и b – от 0 до 10 включительно. Для тестирования функции сравнения двух чисел на равенство a == b необходимы следующие тестовые примеры (тестирование на робастность не проводится):

\*А. a = 3, b = 5, значение функции = false

\*Б. a = 3, b = 3, значение функции = true

В. a = 0, b = 0, значение функции = true

Г. a = 10, b = 10, значение функции = true

\*Д. a = 0, b = 10, значение функции = false

\*Е. a = 10, b = 0, значение функции = false

Пункт 2. Допустимый интервал значений для целых чисел a и b – от 0 до 10 включительно. Для тестирования функции сравнения двух чисел на строгое неравенство a < b необходимы следующие тестовые примеры (тестирование на робастность не проводится):

\*А. a = 5, b = 6, значение функции = true

\*Б. a = 5, b = 4, значение функии = false

\*В. a = 6, b = 6, значение функции = false

Г. a = 0, b = 0, значение функции = false

\*Д. a = 0, b = 10, значение функции = true

\*Е. a = 10, b = 0, значение функции = false

Ж. a = 10, b = 10, значение функции = false

Пункт 3. Допустимый интервал значений для целых чисел a и b – от 0 до 10 включительно. Для тестирования функции сравнения двух чисел на нестрогое неравенство a >= b необходимы следующие тестовые примеры (тестирование на робастность не проводится):

\*А. a = 5, b = 6, значение функции = false

\*Б. a = 5, b = 4, значение функии = true

\*В. a = 6, b = 6, значение функции = true

Г. a = 0, b = 0, значение функции = true

\*Д. a = 0, b = 10, значение функции = false

\*Е. a = 10, b = 0, значение функции = true

Ж. a = 10, b = 10, значение функции = true

**Вопрос 3. Тестирование условных операторов.**

Пункт 1. Необходимо протестировать следующую функцию на соответствие требованию «Функция должна возвращать значение 2, если на вход ей передано значение 3, и значение 3, если на вход ей передано значение 2». Какой из вариантов входных значений выявит несоответствие функции требованиям?

int inverse(int a)

{

if (a == 2) return 3;

else if (a == 3) return 2;

else return 2;

}

А. a = 2

Б. a = 3

\*В. a = 0

\*Г. a = 5

Пункт 2. Необходимо протестировать следующую функцию на соответствие требованию «Функция должна возвращать значение 2, если на вход ей передано значение 3, и значение 3, если на вход ей передано значение 2». Какой из вариантов входных значений выявит несоответствие функции требованиям?

int inverse(int a)

{

return 5-a;

}

А. a = 2

Б. a = 3

\*В. a = 0

\*Г. a = 5

Пункт 3. Необходимо протестировать следующую функцию на соответствие требованию «Функция должна возвращать значение 2, если на вход ей передано значение 3, и значение 3, если на вход ей передано значение 2». Какой из вариантов входных значений выявит несоответствие функции требованиям?

int inverse(int a)

{

if (a == 2) return 3;

else return 2;

}

А. a = 2

Б. a = 3

\*В. a = -2

\*Г. a = 5

**Вопрос 4. Тестирование работы с файловой системой**

Пункт 1. Существует программа, работа которой описывается следующим требованием «Программа должна принимать три параметра – файл, исходный каталог и целевой каталог и копироватьующая указанный файл из исходного каталога в целевой.» Укажите, какие наборы тестовых примеров следует использовать для указания того, что требование описывает поведение программы не полностью

А. файл, исходный каталог и целевой каталог существуют и переданы в качестве параметров

\*Б. файл, исходный каталог и целевой каталог существуют, количество переданных параметров меньше трех

В. файла, исходного и целевого каталога не существует, количество переданных параметров меньше трех

\*Г. файла, исходного и целевого каталога не существует, количество переданных параметров равно трем

Пункт 2. Существует программа, работа которой описывается следующим требованием «Программа должна принимать два параметра – имя файла и имя пользователя, после чего создавать пустой файл, право чтения которого имеет только указанный пользователь.» Укажите, какие наборы тестовых примеров следует использовать для указания того, что требование описывает поведение программы не полностью

А. имя файла и имя пользователя переданы в качестве параметров, пользователь существует в системе.

\*Б. имя файла и имя пользователя переданы в качестве параметров, пользователь не существует в системе.

В. имя файла и имя пользователя переданы в качестве параметров, пользователь существует в системе, на диске отсутствует свободное место

\*Г. в качестве параметров переданы имя файла и имена двух пользователей.

Пункт 3. Существует программа, работа которой описывается следующим требованием «Программа должна принимать два параметра – два имени файла, после чего создавать копию первого файла с именем второго файла» Укажите, какие наборы тестовых примеров следует использовать для указания того, что требование описывает поведение программы не полностью

А. имена файлов переданы в качестве параметров, файла со вторым именем не существует.

\*Б. имена файлов переданы в качестве параметров, до запуска программы уже существует файл со вторым именем.

\*В. имена файлов переданы в качестве параметров, файлы существуют, на диске отсутствует свободное место

\*Г. в качестве параметров переданы имена трех файлов

**Вопрос 5. Тест-требования и тест-планы**

Пункт 1. Выберите верные утверждения

\*А. тест-планы пишутся на основе тест-требований

Б. тест-планы определяют последовательность действий тестировщика, необходимых для выполнения каждого тестового примера

\*В. тест-план состоит из определений тестовых примеров

\*Г. тестовый пример обязательно содержит описание ожидаемой реакции системы

Пункт 2. Выберите верные утверждения

\*А. тест-план определяет конкретный сценарий проведения тестов

\*Б. тест-требования определяют, какая функциональность системы должна быть протестирована

В. количество тестовых примеров в тест-плане должно быть равно количеству тест-требований

Г. тест-требования всегда содержат определения входных значений, передаваемых системе в ходе тестирования

Пункт 3. Выберите верные утверждения

А. наличие дефекта в системе никогда не обнаруживается при тестировании при помощи корректных данных

Б. вместо проверки на граничных условиях с той же эффективностью можно использовать проверку за границами допустимого интервала

\*В. при тестировании на корректных данных требуется определять один тестовый пример для каждого класса эквивалентности

Г. при тестировании на некорректных данных все некорректные данные могут быть собраны в один класс эквивалентности

**Вопрос 6. Классы эквивалентности**

Пункт 1. Выберите классы эквивалентности для имени файла:

\*А. содержит только буквы и цифры

\*Б. содержит спец-символы

\*В. содержит пробелы

\*Г. задано в виде полного или краткого пути

\*Д. имеет длину больше максимально возможной в операционной системе

Пункт 2. Выберите классы эквивалентности для треугольника, заданного своими сторонами:

\*А. треугольник с нулевой длиной сторон

\*Б. треугольник, у которого сумма двух сторон равна третьей

\*В. треугольник, у которого сумма двух сторон больше третьей

Г. прямоугольный равнобедренный треугольник

Пункт 3. Выберите классы эквивалентности для изображения, загружаемого в память

\*А. изображение нулевого размера

\*Б. изображение, размер которого равен размеру свободной памяти

\*В. изображение, размер которого равен размеру файла подкачки

\*Г. изображение, размер которого превышает размер доступной памяти (физической и виртуальной)

**Вопрос 7. Классы эквивалентности**

Пункт 1. Укажите на проблемы, которые существуют при тестировании граничных диапазонов у чисел с плавающей запятой

\*А. неопределенность самого близкого к предельному значения

Б. слишком большие интервалы значений

В. невозможность использования операций сравнения в предельных случаях

\*Г. невозможность точно присвоить значения, близкие к предельным из-за органичений внутреннего представления чисел

Пункт 2. Укажите основные причины тестирования классов эквивалентности

\*А. обобщение результатов тестирования на большое число различных входных данных

\*Б. снижение количества тестовых примеров

В. проведение автоматизированного полного перебора всех входных значений вместо ручного

Г. сохранение результатов тестирования при изменении классов эквивалентности

Пункт 3. Укажите основные критерии выбора классов эквивалентности

\*А. В один класс попадают все входные значения, для которых одинаково выходное значение системы

Б.Классы эквивалентности выбираются исходя из ограничений, которые накладываются на систему языком программирования

\*В. Всегда существует как минимум 2 класса эквивалентности

Г. В каждом классе эквивалентности должно быть более одного возможного варианта входного значения

**Вопрос 8. Тест-требования**

Пункт 1. Укажите требования, которые невозможно протестировать при помощи тестовых примеров

\*А. Проверить, что программа работает правильно

\*Б. Проверить, что создаваемые файлы имеют нужные права доступа

В. Проверить, что если программа вызвана менее чем с тремя параметрами командной строки, выводится сообщение об ошибке «Недостаточно параметров»

Г. Проверить, что если программа работает дольше 5 секунд, то на экран выводится сообщение «Подождите»

Пункт 2. Укажите требования, которые невозможно протестировать при помощи тестовых примеров

А. Проверить, что программа складывает два числа в соответствии с правилами десятичной арифметики

\*Б. Проверить, что программа имеет удобный интерфейс пользователя

В. Проверить, что при отсутствии файла конфигурации программа запускается с настройками по умолчанию

\*Г. Проверить, что программа работает достаточно быстро

Пункт 3. Укажите требования, которые невозможно протестировать при помощи тестовых примеров

А. Проверить, что в случае отсутствия прав доступа на запись к выходному файлу, программа завершает свое выполнение с кодом ошибки 12.

\*Б. Проверить, что сочетания клавиш, используемые в программе, интуитивно понятны

В. Проверить, что сообщения об ошибках, выдаваемые программой выводятся красным цветом в диалоговом окне с заголовком «Ошибка»

\*Г. Проверить, что программу невозможно завершить

**Раздел. «Анализ покрытия программного кода»**

**Вопрос 1. Покрытие программного кода**

Пункт 1 Укажите реально существующие типы покрытия программного кода

\*А. по строкам

\*Б. по компонентам логических выражений

\*В. по ветвям

Г. по скобкам

Пункт 2 Что необходимо для полного покрытия по MC/DC?

\*А. каждое логическое условие должно принимать все возможные значения

\*Б. каждая компонента логического условия должна хотя бы один раз принимать все возможные значения

\*В. должно быть установлено независимое влияние каждой из переменных на результат

Г. каждая компонента логического условия должна быть протестирована независимо от остальных

Пункт 3. Укажите правильные последовательности анализа покрытий

\*А. покрытие требований, покрытие кода по строкам, покрытие кода по MC/DC

Б. покрытие по MC/DC, покрытие по ветвям, покрытие по строкам

В. покрытие по ветвям, покрытие по строкам, покрытие требований

\*Г. покрытие требований, покрытие по MC/DC

**Вопрос 2. Тест-планы**

Пункт 1. Выберите верные свойства тест-плана

\*А. независимость тестовых примеров

\*Б. единая схема идентификации и записи тестовых примеров

В. независимость тестового окружения

Г. определение тестов на формальном языке

Пункт 2. Выберите типичные компоненты определения тестового примера в тест-плане

\*А. идентификатор тестового примера

\*Б. входные данные

\*В. ожидаемые выходные данные

\*Г. описание теста

Пункт 3. Какие проблемы могут существовать в корректных по формату тест-планах?

\*А. запутанная идентификация и последовательность тестовых примеров

\*Б. зависимости между тестовыми примерами

\*В. недостаточная подробность описания сценария теста

Г. отсутствие ожидаемого выходного результата

**Вопрос 3. Отчеты о прохождении тестов**

Пункт 1. В состав статистики о прохождении тестов может входить следующая информация:

А. количество непокрытых строк исходного кода

\*Б. количество пройденных и не пройденных тестовых примеров

\*В. не совпавшие ожидаемые и реальные выходные данные

\*Г. идентификаторы выполненных тестовых примеров

Пункт 2. На что может указывать не прошедший тестовый пример?

\*А. на дефект в требованиях

\*Б. на дефект в коде

\*В. на дефект в тестовом драйвере

\*Г.на дефект в тестовом примере

Пункт 3. Если при изменении программного кода системы повторное выполнение тестов выявляет не прошедшие тестовые примеры, то причина может быть в

\*А. изменившихся требованиях

\*Б. изменившемся коде

В. изменившемся тестовом окружении

\*Г. выявлении ранее замаскированных старых дефектов

**Вопрос 4. Покрытие программного кода по строкам**

Пункт 1. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по строкам?

if ( (a == 0) && (b == 0)) {

call\_1();

} else {

call\_2();

}

\*А. 1) a = 0, b = 0; 2) a = 1, b = 1

\*Б. 1) a = 1, b = 0; 2) a = 0, b = 0

В. 1) a = 0, b = 1; 2) a = 1, b = 0

Г. 1) a = 1, b = 1; 2) a = 1, b = 0

Пункт 2. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по строкам?

if (a == 0) {

if (b == 0) call\_1();

} else {

call\_2();

}

А. 1) a = 0, b = 0; 2) a = 0, b = 1; 3) a = 1; b = 0

\*Б. 1) a = 0, b = 0; 2) a = 1, b = 0

\*В. 1) a = 0, b = 0; 2) a = 1, b = 1

Г. 1) a = 0, b = 1; 2) a = 1, b = 0

Пункт 3. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по строкам?

if (a == 0) {

call\_1();

} else {

if (b > 0) call\_2();

}

А. 1) a = 0, b = 0; 2) a = 0, b = 1; 3) a = 1; b = 0

\*Б. 1) a = 0, b = 0; 2) a = 1, b = 1

\*В. 1) a = 0, b = 1; 2) a = 1, b = 1

Г. 1) a = 0, b = 1; 2) a = 1, b = 0

**Вопрос 5. Покрытие программного кода по ветвям**

Пункт 1. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по ветвям?

if ( (a == 0) && (b == 0)) {

call\_1();

} else {

call\_2();

}

\*А. 1) a = 0, b = 0; 2) a = 1, b = 1

\*Б. 1) a = 1, b = 0; 2) a = 0, b = 0

В. 1) a = 0, b = 1; 2) a = 1, b = 0

Г. 1) a = 1, b = 1; 2) a = 1, b = 0

Пункт 2. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по ветвям?

if (a == 0) {

if (b == 0) call\_1();

} else {

call\_2();

}

А. 1) a = 0, b = 0; 2) a = 0, b = 1; 3) a = 1; b = 0

\*Б. 1) a = 0, b = 0; 2) a = 1, b = 0; 3) a = 0, b = 1

В. 1) a = 0, b = 0; 2) a = 1, b = 1; 3) a = 1, b = 0

Г. 1) a = 0, b = 1; 2) a = 1, b = 0

Пункт 3. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по ветвям?

if (a == 0) {

call\_1();

} else {

if (b > 0) call\_2();

}

\*А. 1) a = 0, b = 0; 2) a = 1, b = 1; 3) a = 1; b = 0

Б. 1) a = 0, b = 0; 2) a = 1, b = 1

В. 1) a = 0, b = 1; 2) a = 0, b = 0; 3) a = 1; b = 1

Г. 1) a = 0, b = 1; 2) a = 0, b = 1; 3) a = 0; b = 0

**Вопрос 6. Покрытие программного кода по условиям**

Пункт 1. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по cтрокам и условиям?

if ( (a == 0) || (b = 0) || (c == 1) {

call\_1();

} else {

if (d == 1) call\_2()

}

А. 1) a = 0, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0

\*Б. 1) a = 0, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0; 2) a = 1, b = 1, c = 0, d = 1

\*В. 1) a = 1, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0; 2) a = 1, b = 1, c = 0, d = 1

Г. 1) a = 0, b = 0, c = 1, d = 0 ; 2) a = 1, b = 1, c = 0, d = 0; 2) a = 1, b = 1, c = 0, d = 0

Пункт 2. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по cтрокам и условиям?

if ( (a == 0) && (b = 0) && (c == 1) {

call\_1();

} else {

if (d == 1) call\_2()

}

А. 1) a = 0, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0

\*Б. 1) a = 0, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0; 2) a = 1, b = 1, c = 0, d = 1

В. 1) a = 1, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0; 2) a = 1, b = 1, c = 0, d = 1

Г. 1) a = 0, b = 0, c = 1, d = 0 ; 2) a = 1, b = 1, c = 0, d = 0; 2) a = 1, b = 1, c = 0, d = 0

Пункт 3. Какие минимальные наборы тестовых примеров можно использовать для полного покрытия следующего участка программного кода по cтрокам и условиям?

if ( (a == 0) && (b = 0) && (c == 1) {

if (d == 1) call\_1();

} else {

call\_2()

}

А. 1) a = 0, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0

\*Б. 1) a = 0, b = 0, c = 1, d = 1 ; 2) a = 0, b = 0, c = 1, d = 0; 2) a = 1, b = 1, c = 0, d = 1

В. 1) a = 1, b = 0, c = 1, d = 1 ; 2) a = 1, b = 1, c = 0, d = 0; 2) a = 1, b = 1, c = 0, d = 1

Г. 1) a = 0, b = 0, c = 1, d = 0 ; 2) a = 1, b = 1, c = 0, d = 0; 2) a = 1, b = 1, c = 0, d = 0

**Вопрос 7. Покрытие программного кода по MC/DC**

Пункт 1. Сколько нужно тестовых примеров для покрытия следующего участка программного кода по MC/DC?

if ( (a == 0) || (b = 0) || (c == 1) {

call\_1();

} else {

call\_2()

}

А. 3

Б. 5

\*В. 4

Г. 2

Пункт 2. Сколько нужно тестовых примеров для покрытия следующего участка программного кода по MC/DC?

if ( (a == 0) && (b = 0) && (c == 1) {

call\_1();

} else {

call\_2()

}

А. 3

Б. 5

\*В. 4

Г. 2

Пункт 2. Сколько нужно тестовых примеров для покрытия следующего участка программного кода по MC/DC?

if ( !(a == 0) && (b = 0) && (c == 1) {

call\_1();

} else {

call\_2()

}

А. 3

Б. 5

\*В. 4

Г. 2

**Вопрос 8. Анализ покрытия**

Пункт 1. Какие проблемы может выявить анализ полноты покрытия тестами кода?

\*А. неполноту тестовых процедур

\*Б. неполноту требований

\*В. никогда не выполняемый код

Г. неполноту программного кода

Пункт 2.Какие процедуры могут применяться тестировщиком для улучшения покрытия?

А. уточнение требований

Б. устранение никогда не выполняемого кода

\*В. написание дополнительных тестовых примеров

Г. изменение уровня покрытия

Пункт 3.Какие процедуры могут применяться разработчиком для улучшения покрытия?

\*А. уточнение требований

\*Б. устранение никогда не выполняемого кода

В. написание дополнительных тестовых примеров

Г. изменение уровня покрытия

**Раздел. «Повторяемость тест планов»**

**Вопрос 1. Регрессионное тестирование**

Пункт 1 Для чего проводят регрессионное тестирование?

\*А. для проверки того, что изменения, внесенные в один из компонентов системы не нарушили логику работы других ее компонент

\*Б.для проверки того, что все тесты остались работоспособными на новой версии системы

В. для определения того, какие тесты можно больше никогда не выполнять

Г. для выявления не обнаруженных ранее дефектов системы при помощи новых тестов

Пункт 2 Какие результаты могут быть получены в результате регрессионного тестирования?

\*А. Все тесты пройдены успешно

\*Б. Некоторые тесты невозможно корректно выполнить

\*В. Некоторые тесты не пройдены

\*Г. Все тесты не пройдены

Пункт 3. Какие действия может понадобиться применить после неуспешного прохождения некоторых старых тестов при регрессионном тестировании?

\*А. Удалить не пройденные тесты

Б. Скорректировать новые тесты

\*В. Скорректировать программный код системы так, чтобы старые тесты проходили

\*Г. Скорректировать старые тесты так, чтобы они соответствовали новой логике работы системы

**Вопрос 2. Предусловия для выполнения тестов**

Пункт 1. Что из перечисленного ниже может служить предусловием для выполнения теста?

\*А. Объем доступного дискового пространства

\*Б. Значения входных переменных

В. Значения выходных переменных текущего теста

\*Г. Значения выходных переменных, установленных в предыдущем тесте

Пункт 2. Что из перечисленного ниже может служить предусловием для выполнения теста?

\*А. Наличие или отсутствие определенных файлов на диске

Б. Невозможность запустить тестовое окружение

\*В. Состояние тестового окружения до момента запуска теста

Г. Состояние тестового окружения во время работы предыдущего теста

Пункт 3. Что из перечисленного ниже может служить предусловием для выполнения теста?

\*А. Объем свободного дискового пространства

\*Б. Разрешение экрана, при котором выполняется программа

В. Количество строк программного кода в тестируемой программе

Г. Квалификация пользователя, работающего с программой

**Вопрос 3. Предусловия для выполнения тестов**

Пункт 1. Выберите корректные предусловия для тестов, который проверят следующее тест-требование: «Проверить, что при создании файла с именем OUT.TXT первой строкой в файл записывается текущая дата и время».

\*А. Объем свободного дискового пространства достаточен для записи файла на диск

\*Б. Объем свободной оперативной памяти достаточен для хранения информации о дате и времени

\*В. Файл с именем OUT.TXT не существует перед выполнением теста

\*Г. Текущее время не равно 00:00

Пункт 2. Выберите корректные предусловия для тестов, который проверят следующее тест-требование: «Проверить, что программа копирует все файлы из каталога DIR1 в каталог DIR2».

\*А. Объем свободного дискового пространства достаточен для записи файлов на диск

\*Б. Каталоги DIR1 и DIR2 не существуют

\*В. Каталог DIR2 не доступен для записи в него файлов

\*Г. В каталоге DIR1 отсутствуют файлы

Пункт 3. Выберите корректные предусловия для тестов, который проверят следующее тест-требование: «Проверить, что программа выводит на экран список всех файлов из каталога DIR1».

\*А. Вывод на экран разрешен

\*Б. Каталог DIR1 существуют

\*В. Каталог DIR1 не доступен для чтения списка его файлов

\*Г. В каталоге DIR1 отсутствуют файлы

**Вопрос 4. Зависимости между тестовыми примерами**

Пункт 1. Выберите верные утверждения

А. Зависимости между тестовыми примерами полностью недопустимы

\*Б. При наличии зависимостей между тестовыми примерами затруднен процесс выборочного регрессионного тестирования

\*В. При наличии зависимостей между тестовыми примерами может быть сложно изменить порядок следования тестов

Г. При наличии зависимостей между тестовыми примерами легче отслеживать изменения в системе

Пункт 2. Выберите верные утверждения

\*А. Зависимости между тестовыми примерами могут помочь снизить сложность тестовых сценариев

Б. Зависимости между тестовыми примерами возникают при неоднозначном состоянии тестового окружения на момент начала теста

В. Случайные зависимости между тестовыми примерами сложно выявлять

\*Г. Можно обезопасить себя от появления зависимостей между тестовыми примерами, если перед началом выполнения каждого теста полностью переинициализировать тестовое окружение

Пункт 3. Выберите верные утверждения

А. Зависимости между тестовыми примерами могут снизить общее время выполнения тестов и не ухудшают сопровождаемость тестовых примеров.

\*Б. Зависимости между тестовыми примерами могут снизить общее время выполнения тестов, но ухудшают сопровождаемость тестовых примеров

\*В. Зависимости между тестовыми примерами являются следствием неполной инициализации тестового окружения перед началом выполнения тестового примера

\*Г. Для обнаружения зависимостей между тестовыми примерами можно менять порядок выполнения тестовых примеров.

**Вопрос 5. Регрессионное тестирование**

Пункт 1. Какая документация должна быть проанализирована при неуспешном прохождении регрессионного тестирования?

\*А. Старые тест-требования

\*Б. Новые тест-требования

\*В. Не изменившиеся тест-требования

\*Г. Удаленные тест-требования

Пункт 2. В каких случаях регрессионное тестирование может проводиться вручную?

\*А. При небольшом количестве тестовых примеров

Б. При необходимости регулярного повторения тестовых примеров

В. При необходимости ручного анализа результатов тестирования

Г. При наличии зависимостей между тестовыми примерами

Пункт 3. Какие типы тестов не рекомендуется применять в ходе регрессионного тестирования?

А. Автоматические тесты

\*Б. Тесты с зависимостями от других тестов

\*В. Ручные тесты

Г. Старые тесты

**Вопрос 6. Зависимости между тестовыми примерами**

Пункт 1. В каких случаях может быть выявлена зависимость между тестовыми примерами?

\*А. При изменении порядка выполнения тестов

\*Б. При изменении набора выполняемых тестов

\*В. При изменении начальных условий тестов

Г. При изменении ожидаемых значений тестов

Пункт 2. Укажите возможные способы устранения зависимости между тестовыми примерами

\*А. Полная реинициализация тестового окружения перед выполнением тестов

\*Б. Выполнение тестов по одному за раз

В. Изменение порядка следования тестовых примеров

Г. Отслеживание состояния памяти системы в ходе тестирования

Пункт 3. В каких случаях зависимость между тестовыми примерами может остаться не выявленной?

\*А. Система не меняется значительное время

\*Б. Не проводится частичное регрессионное тестирование

\*В. Тестовые примеры выполняются всегда в одном и том же порядке

Г. Тестовые примеры выполняются всегда по одному

**Вопрос 7. Зависимости между тестовыми примерами**

Пункт 1. Что может являться причиной появления зависимостей между тестовыми примерами?

\*А. Использование настроек по умолчанию для групп тестов

Б. Использование общей памяти

\*В. Недостаточная очистка памяти после выполнения тестового примера

Г. Недостаточный уровень покрытия предыдущими тестами кода

Пункт 2.В каких случаях зависимость между тестовыми примерами полезна?

\*А. В случае, если один тестовый пример выполняет всю инициализацию тестового окружения и тестируемой системы, а все зависимые примеры начинают работу, считая, что система уже инициализирована

\*Б. В случае, если тестируется последовательное изменение состояний системы

В. В случае, если тестируются параллельные вычисления

Г. В случае, если тестируются большое количество разнородных требований

Пункт 3. Укажите возможные способы выявления данных, приводящих к зависимостям тестовых примеров

\*А. Выделять выходные переменные, которые не менялись в ходе выполнения теста

Б. Выделять маршруты выполнения тестов

В. Выделять входные переменные, инициализированные только в одном тесте

Г. Выделять не покрытые участки кода

**Вопрос 8. Риски регрессионного тестирования и зависимостей между тестовыми примерами**

Пункт 1. Какие риски и причины рисков свойственны процессу проведения регрессионного тестирования?

\*А. Выполнение регрессионного набора тестов прерывается в середине процесса

\*Б. Набор тестов для регрессионного тестирования не соответствует тестируемой версии системы

\*В. Регрессионное тестирование требует значительного времени на проведение

Г. Регрессионное тестирование не выявляет никаких дефектов

Пункт 2. Какие риски и причины рисков существуют при наличии зависимостей между тестовыми примерами?

\*А. Невозможность произвольно выбирать тесты для выполнения

\*Б. Затруднения при выделении выборочных наборов тестов для регрессионного тестирования

\*В. Маскировка дефектов при изменениях системы

\*Г. Усложнение анализа и модификации тестовых примеров

Пункт 3. Какие риски и причины рисков существуют при неверном определении предусловий тестовых примеров?

\*А. Не полностью определены все возможные варианты предусловий

\*Б. Невозможность создать заданные предусловия на тестовом стенде

В. Невозможность создать общие процедуры для создания среды тестирования, заданной в предусловии

Г. Не возникают зависимости между тестовыми примерами

**Раздел. «Документация.** **Тест требования»**

**Вопрос 1. Документация, сопровождающая процесс тестирования**

Пункт 1 Укажите типы документов, непосредственно сопровождающих процесс тестирования

\*А. отчеты о покрытии

Б.планы конфигурации

\*В. отчеты о проблемах

\*Г. тест-требования

Пункт 2 Укажите типы документов, непосредственно сопровождающих процесс тестирования

А. низкоуровневые требования

\*Б.план тестирования

\*В. отчеты о результатах выполнения тестов

Г. планы гарантии качества

Пункт 3 Укажите типы документов, непосредственно сопровождающих процесс тестирования

\*А. низкоуровневые тест-требования

\*Б. запросы на изменение

\*В. планы верификации архитектуры

\*Г. план полунатурных испытаний

**Вопрос 2. План тестирования**

Пункт 1 Укажите основные задачи плана тестирования

\*А. определение границ тестирования

Б.определение тестовых примеров

\*В. определение плана-графика тестирования

\*Г. определение общего подхода к тестированию

Пункт 2 Укажите элементы, которые рекомендуется включать в план тестирования

А. определения тестовых примеров

\*Б.требования к среде тестирования

В. предусловия для выполнения каждого тестового примера

\*Г. риски

Пункт 3 Укажите элементы, которые рекомендуется включать в план тестирования

\*А. определение тестируемых элементов системы

\*Б. критерий успешности/неуспешности тестирования

\*В. требования к уровню подготовки тестировщиков

Г. описания сценариев тестов

**Вопрос 3. Тест-требования**

Пункт 1 Укажите основные свойства совокупности тест-требований к части системы

\*А. Полнота

\*Б. Непротиворечивость

\*В. Тестируемость

Г. Замкнутость

Пункт 2 Укажите основные группы тест-требований

\*А. тест-требования к контролю входных данных

\*Б. тест-требования к основному сценарию работы системы

В. тест-требования к основному сценарию работы драйвера

Г. тест-требования к контролю выходных результатов тестов

Пункт 3 Укажите основные группы тест-требований

\*А. тест-требования к обработке исключительных ситуаций

\*Б. тест-требования к процессу вывода результатов

В. тест-требования к работе с запросами на изменение

Г. тест-требования к работе с данными тестирования

**Вопрос 4. Тест-требования**

Пункт 1. Каким свойством не обладает требование «Проверить, что система работает правильно»?

А. Полноты

\*Б. Тестируемости

В. Непротиворечивости

Г. Замкнутости

Пункт 2. Каким свойством не обладает требование «Проверить, что пароль пользователя имеет длину не менее 2 символов»?

\*А. Полноты

Б. Тестируемости

В. Непротиворечивости

Г. Замкнутости

Пункт 3. Каким свойством не обладает требование «Проверить, что система реализует все требования пользователя»?

А. Полноты

\*Б. Тестируемости

В. Непротиворечивости

Г. Замкнутости

**Вопрос 5. Документация, сопровождающая процесс тестирования**

Пункт 1. Укажите правильную последовательность создания документов

\*А. Тест-требования -> Тест-планы -> Отчеты о тестировании -> Запросы на изменение

Б. Тест-планы -> Тест-требования –> Запросы на изменение

В. Тест-планы -> Запросы на изменение -> Отчеты о тестировании

\*Г. Функциональные требования -> Тест-требования

Пункт 2. Без каких документов можно обойтись при создании тест-планов для функционального тестирования?

\*А. Тест-требования

Б. Функциональные требования

\*Г. Архитектура

\*Д. Низкоуровневые тест-требования

Пункт 3. На какие документы иногда разбивают план верификации?

\*А. План верификации системных требований

\*Б. План верификации архитектуры

\*В. План системного тестирования

\*Г. План нагрузочного тестирования

**Вопрос 6. Тест-требования**

Пункт 1. Укажите причины появления нетестируемых тест-требований

\*А. Недостаточное представление автора требований о процессе тестирования

Б. Невозможность сформулировать требование иначе

\*В. При написании требования не учитывается критерий успешности прохождения теста на это требование

Г. При написании требования не учитываются реальные потребности пользователя системы

Пункт 2. Укажите причины появления неполных (по отношению к функциональным) тест-требований

\*А. Недостаточная степень знакомства автора тест-требований с функциональными требованиями

\*Б. Изменение функциональных требований, которое не повлекло за собой изменение тест-требований

В. Сужение границ тестирования

Г. Недостаточная степень детализации функциональных требований

Пункт 3. Укажите причины появления противоречивых тест-требований

\*А. Недостаточная степень знакомства автора тест-требований с функциональными требованиями

\*Б. Изменение функциональных требований, которое не повлекло за собой изменение тест-требований

В. Сужение границ тестирования

Г. Недостаточная степень детализации функциональных требований

**Вопрос 7. Тест-требования**

Пункт 1. Какие из приведенных ниже фраз нельзя считать верифицируемыми тест-требованиями? Считать, что верификация проводится методом черного ящика.

А. «Проверить, что система работает в соответствии с функциональным требованием 4.3.5.1»

Б. «Проверить, что интерфейс системы удовлетворяет критерию интуитивной понятности, определенном в документе DEFG-12334»

\*В. «Проверить, что система завершает свою работу немедленно после своего запуска»

\*Г. «Проверить, что система быстро выдает результат после запроса пользователя»

Пункт 2. Какие из приведенных ниже фраз нельзя считать верифицируемыми тест-требованиями? Считать, что верификация проводится методом черного ящика.

А. «Проверить, что программа копирует из указанного каталога в текущий каталог файлы, расширение имени которых начинается с цифры, заданной параметром N.»

\*Б. «Проверить ввод строк с нечетным количеством символов»

В. «Проверить, что все сообщения выводятся строчными буквами.»

\*Г. «Проверить, что программа очищает память»

Пункт 3. Какие из приведенных ниже фраз нельзя считать верифицируемыми тест-требованиями?

Считать, что верификация проводится методом черного ящика.

\*А. «Проверить, что программа работает в соответствии с руководством пользователя»

\*Б. «Проверить, что программа никогда не выводит сообщений на экран»

В. «Проверить, что в случае вызова программы с тремя параметрами, она завершает свое выполнение»

Г. «Проверить, что в случае, если файл данных не доступен для чтения, программа выводит сообщение об ошибке, определенное в разделе 4.3.5»

**Вопрос 8. Тест-требования**

Пункт 1. Выберите верные утверждения

А. Тест-требования всегда пишутся на основе функциональных требований

\*Б. Существуют низкоуровневые и высокоуровневые тест-требования

В. Тест-требования описывают сценарии тестирования системы

Г. Тест-требования имеют меньшую степень детализации, чем функциональные требования

Пункт 2. Выберите верные утверждения

\*А. Одному функциональному требованию соответствует минимум одно тест-требование

Б. Тест-требования должны обладать свойством полноты, замкнутости и непротиворечивости

\*В. Тест-требования должны обладать свойством полноты, тестируемости и непротиворечивости

Г. Тест-требования должны обладать свойством замкнутости и непротиворечивости

Пункт 3. Выберите верные утверждения

А. При разработке тест-требований необходимо иметь доступ к исходным текстам тестируемой системы

Б. При разработке тест-требований необходимо иметь доступ к функциональным требованиям на тестируемую систему

В. Перед разработкой тест-требований нужно иметь возможность поработать с системой как пользователь

Г. Перед разработкой тест-требований разработка архитектуры системы должна быть полностью завершена

**Раздел. «Документация.** **Тест планы»**

**Вопрос 1. Тест-планы**

Пункт 1 Укажите основные отличия тест-планов от тест-требований

А. Тест-планы служат для создания тестовых сценариев

\*Б. Тест-планы описывают конкретные способы тестирования системы

В. Тест-планы описывают общие подходы к тестированию

Г. Тест-планы пишутся на основе функциональных требований

Пункт 2 Укажите основные критерии качества тест-плана

\*А. Полнота покрытия тест-требований

Б.Наличие подробного описания ожидаемых результатов тестирования

В.Запись тест-плана в формализованной форме

Г.Определение критерия успешного прохождения тестов для каждого тестового примера

Пункт 3. Укажите элементы, входящие в состав описания каждого тестового примера в тест-плане

\*А. Описание тестового сценария

\*Б. Перечень входных воздействий

\*В. Перечень ожидаемых выходных воздействий

\*Г. Ссылка на требование, проверяемое тестовым примером

**Вопрос 2. Тест-планы в виде сценариев**

Пункт 1 Укажите основные недостатки описания тест-планов в виде сценариев

А. Такие тест-планы подходят только для ручного тестирования

Б. Такие тест-планы подходят только для автоматизированного тестирования

В. Такие тест-планы не учитывают предусловия для выполнения группы тестов

Г. Такие тест-планы не предназначены для покрытия по MC/DC

Пункт 2 Укажите основные составные части шага сценария

\*А. Номер шага

Б. Описание действий в ходе шага

\*В. Ссылки на тестируемые требования

\*Г. Описание ожидаемых результатов шага

Пункт 3 Чем отличаются сценарии для автоматического тестирования от сценариев для ручного?

\*А. Они записаны на формальном языке

Б. Они не включают в себя критерий успешности/неуспешности прохождения теста

\*В. Они не могут быть совмещены с отчетами о прохождении тестов

Г. Они не могут быть выполнены вручную

**Вопрос 3. Тест-планы в виде таблиц**

Пункт 1. Укажите недостатки табличного представления тест-планов

А. Большой размер таблицы, затрудняющий анализ тест-планов

Б. Невозможность задать предусловия для тестовых примеров

В. Невозможность преобразовать к представлению в виде сценария

Г. Невозможность выполнения ручного тестирования при таком представлении

Пункт 2. Укажите, какие варианты представления возможны для задания тест-плана в виде таблицы

\*А. По горизонтали – тестовые примеры, по вертикали – входные переменные и их значения, в ячейках на пересечении примеров и значений – маркеры, означающие использование значения в примере.

\*Б. По горизонтали – тестовые примеры, по вертикали – входные переменные, в ячейках на пересечении – значения

\*В. По горизонтали – значения входных переменных, по вертикали – имена входных переменных, на пересечении – номера тестовых примеров

\*Г. По горизонтали – значения входных переменных, по вертикали – номера тестовых примеров, на пересечении – имена входных переменных

Пункт 3. В каких случаях рекомендуется применять табличное представление тест-планов?

А. Большое количество разнородных входных данных

\*Б. Большое количество однородных входных данных

\*В. Большое количество входов системы

Г. Большое количество предусловий тестов

**Вопрос 4. Тест-планы в виде конечных автоматов**

Пункт 1. Из чего состоит тест-план в виде описания конечного автомата?

А. Из описания сценариев тестовых примеров

\*Б. Из описания состояний

\*В. Из описания сценариев переходов

\*Г. Из описания протокола обмена

Пункт 2. Какой тип покрытия более полно проверяет работу конечного автомата?

А. Автомат побывал во всех состояниях

\*Б. Автомат совершил все возможные переходы

В. Автомат передал все возможные данные

Г. Автомат вернулся в исходное состояние

Пункт 3. Укажите недостатки представления тест-планов в виде конечных автоматов

А. Предназначены для тестирования последовательного обмена данными между системами

Б. Невозможно определить реакцию на непредусмотренное событие

В. Невозможность автоматического преобразования в сценарии

Г. Невозможность определения степени покрытия таких тестов

**Вопрос 5. Отчеты о прохождении тестов**

Пункт 1. Какая информация включается в отчет о прохождении тестов?

\*А. Идентификаторы тестовых примеров

\*Б. Результат выполнения тестов

\*В. Общая статистика выполнения тестов

Г. Степень покрытия тестами требований

Пункт 2. Для чего нужны отчеты о прохождении тестов?

\*А. Для определения некорректно реализованных требований

Б. Для определения точного местоположения дефекта в программном коде

\*В. Для определения количества дефектов, оставшихся в системе

Г. Для предотвращения появления дефектов в программной системе

Пункт 3. При помощи какой информации можно сделать заключение о необходимости повторного тестирования?

\*А. По наличию неуспешно пройденных тестов

\*Б. По наличию тестов, которые не могли быть запущены

В. По наличию успешно пройденных тестов

\*Г. По наличию тестов, выполнение которых прервалось аварийно

**Вопрос 6. Отчеты о прохождении ручных тестов**

Пункт 1. Какая информация заносится тестировщиком в отчет о прохождении тестов в процессе ручного тестирования?

А. Номер тестового примера

В. Ожидаемые выходные значения

\*Г. Реальные выходные значения

\*Д. Отметка о т ом, пройден тест или не пройден

Пункт 2. Какая форма представления тест-плана не может быть использована для ручного тестирования?

А. В виде конечного автомата

Б. Табличная

В. В виде сценариев

Г. Автоматически сгенерированная

Пункт 3. Выберите верное утверждение

А. Отчет о ручном тестировании не может служить источником данных для создания запроса на изменение

\*Б. Результаты ручного тестирования, проведенного разными тестировщиками, могут различаться

В. Отчет о ручном тестировании может храниться только в бумажной форме

\*Г. Возможно использование формального языка для отметок в отчете о ручном тестировании

**Вопрос 7. Упрощение тест-планов**

Пункт 1**.** В каких случаях возможно упросить тест-план, исключив из него отдельные тестовые примеры, не снизив при этом уровня покрытия системы?

\*А. Удаляются тесты для уже не существующей функциональности

\*Б. Удаляются тесты, дублируемые другими тестами

В. Удаляются тесты, выполнение которых занимает слишком много времени

Г. Удаляются тесты, выполнение которых требует слишком много вычислительных ресурсов

Пункт 2.В каких случаях не рекомендуется упрощать тест-планы?

\*А. Если в проект привлекается много новых сотрудников

\*Б. Если тестируемая система относится к разряду критических или подвергается стороннему аудиту

В. Если упрощения позволяют ускорить создание тестовых примеров

\*Г. Если упрощения вызывают неточности в задании входных данных

Пункт 3. Какие действия приведут к ухудшению качества тестирования системы? Будем считать, что тесты в тест-плане до и после упрощения дают приемлемый уровень покрытия.

\*А. Упрощение тест-планов и снижение приемлемого уровня покрытия

Б. Упрощение тест-планов без снижения приемлемого уровня покрытия

\*В. Снижение приемлемого уровня покрытия без упрощения тест-планов

Г. Упрощение тест-планов и повышение приемлемого уровня покрытия

**Вопрос 8. Тест-планы в виде конечных автоматов**

Пункт 1. Какие проблемы могут быть выявлены при помощи тест-плана в виде конечного автомата?

\*А. Состояние отсутствует в системе

\*Б. Переход отсутствует в системе

\*В. Система не реагирует на определенные сообщения

\*Г. Система переходит в неверное состояние

Пункт 2. Какие проблемы не могут быть выявлены при помощи тест-плана в виде конечного автомата? (Предположим, что тестирование проводится методом «черного ящика»)

\*А. Наличие в системе дополнительных состояний, не указанных в требованиях

Б. В процессе перехода между состояниями выполняются неверные функции

\*В. Наличие в системе дополнительных переходов, не указанных в требованиях

Г. В системе отсутствуют переходы, указанные в требованиях

Пункт 3. Какие проблемы могут возникнуть при несовпадении протоколов взаимодействия тестируемого автомата и тестирующего автомата?

\*А. Автоматы не уйдут из начального состояния

\*Б. Автоматы остановятся в середине выполнения теста

В. Автоматы не смогут принять и передать сообщения

Г. Автоматы сразу перейдут в конечное состояние

**Раздел. «Документация.** **Отчеты»**

**Вопрос 1. Отчеты о покрытии**

Пункт 1 В какой момент времени могут быть сгенерированы отчеты о покрытии

\*А. Параллельно с созданием отчета о прохождении тестов

\*Б.После создания отчета о прохождении тестов

В. После первичной инициализации тестового окружения

Г. После выполнения каждого тестового примера

Пункт 2. О чем может свидетельствовать неполное покрытие программного кода тестами?

А. О избыточной сложности тестируемой системы

\*Б. О неполноте тестов

\*В. О том, что допустимый уровень покрытия меньше 100%

\*Г. О наличии участков защитного или мертвого кода

Пункт 3. Какие изменения может потребоваться внести в систему по результатам анализа отчета о покрытии?

\*А. Дополнить требования

\*Б. Удалить неиспользуемые участки программного кода

В. Добавить новый код

\*Г. Добавить новые тесты

**Вопрос 2. Анализ отчетов о покрытии**

Пункт 1. Какая информация должна содержаться в отчете о покрытии и стандартах проекта для определения того, что уровень покрытия достаточен?

\*А. Тип покрытия

\*Б. Критерий достаточности покрытия для его различных типов

\*В. Степень покрытия функций модуля

Г. Список функций модуля

Пункт 2. Какая информация должна содержаться в отчете о покрытии и стандартах проекта для определения того, что уровень покрытия требований недостаточен?

А. Тип покрытия

Б. Критерий достаточности покрытия для его различных типов

\*В. Количество покрытых требований

\*Г. Общее количество требований

Пункт 3. Что необходимо предпринять тестировщику после выявления недостаточного покрытия в модуле?

\*А. Изменить тесты, которые привели к недостаточному покрытию

\*Б. Создать запрос на изменение/отчет о проблеме, в котором изложить свой взгляд на причины недопокрытия

В. Изменить требования, по которым были составлены тесты

Г. Изменить критерий достаточности покрытия

**Вопрос 3. Анализ отчетов о покрытии**

Пункт 1. По каким причинам может изменяться степень покрытия программного кода тестируемого модуля, если программный код не меняется?

\*А. Из-за оптимизации компилятора

Б. Из-за длительности тестирования

\*В. В зависимости от используемого набора тестов

\*Г. Из-за возникновения зависимости между тестовыми примерами

Пункт 2. В каких случаях при обнаружении недостаточного покрытия не нужно предпринимать никаких действий по изменению тестов или кода?

\*А. Непокрытый код оставлен «на будущее» и никак не используется в текущей версии системы

\*Б. Непокрытый код представляет собой обработчики исключительных ситуаций, которые не могут быть сгенерированы тестовым окружением

В. Непокрытый код не является существенным с точки зрения разработчика

Г. Непокрытый код перестал выполняться после выпуска новой версии системы, но на старой покрывался

Пункт 3. В какой последовательности необходимо проводить анализ покрытия?

А. покрытие кода -> покрытие требований -> покрытие тестов

Б. покрытие требований -> покрытие кода -> покрытие тестов

В. покрытие требований -> покрытие кода

Г. покрытие кода -> покрытие требований

**Вопрос 4. Отчеты о покрытии исходного кода**

Пункт 1. Если при создании отчета о покрытии исходного кода указано, что уровень покрытия по ветвям – 100%, то про какие уровни покрытия можно утверждать, что их уровень покрытия также 100%?

А. По инструкциям

Б. По MC/DC

В. По компонентам логических условий

Г. По требованиям

Пункт 2. Если при создании отчета о покрытии исходного кода указано, что уровень покрытия по инструкциям – 100%, то про какие уровни покрытия можно утверждать, что их уровень покрытия также 100%?

А. По ветвям

Б. По MC/DC

В. По компонентам логических условий

Г. По требованиям

Пункт 3. Если при создании отчета о покрытии исходного кода указано, что уровень покрытия по MC/DC – 100%, то про какие уровни покрытия можно утверждать, что их уровень покрытия также 100%?

А. По инструкциям

\*Б. По ветвям

\*В. По компонентам логических условий

Г. По требованиям

**Вопрос 5. Отчеты о проблемах**

Пункт 1. По результатам анализа каких документов могут создаваться отчеты о проблемах?

\*А. Руководство пользователя

\*Б. Тест-требования

\*В. Тест-планы

\*Г. Результаты выполнения тестов

Пункт 2. Какими свойствами обязательно должен обладать отчет о проблеме?

\*А. Понятностью другому человеку

\*Б. Достаточной степенью подробности для воспроизведения проблемы

В. Достаточной глубиной анализа причин возникновения проблемы

Г. Ссылками на все типы проектной документации

Пункт 3. Кем могут создаваться отчеты о проблемах?

\*А. Тестировщиками

\*Б. Программистами

\*В. Конечными пользователями

\*Г. Специалистами технической поддержки

**Вопрос 6. Отчеты о проблемах**

Пункт 1. В каких случаях должен создаваться отчет о проблеме?

А. Система ведет себя не в соответствии с ожиданиями пользователя

\*Б. Система ведет себя не в соответствии с руководством пользователя

\*В. Обнаружено противоречие в требованиях на систему

Г. Обнаружен аналогичный отчет о проблеме

Пункт 2. В каких случаях сложно найти причину проблемы по результатам анализа отчетов?

А. Отчетов о проблемах слишком много

\*Б. Отчеты о проблемах противоречат друг другу

\*В. Проблему невозможно воспроизвести

Г. Отчеты о проблемах дублируют друг друга

Пункт 3. В каких случаях можно считать, что проблема, описанная в отчете, успешно разрешена и не требуется проводить дальнейшие работы по ее устранению?

\*А. Проблему более не удается воспроизвести

Б. Описание проблемы в отчете слишком нечеткое

В. Проблема имеет слишком низкую степень важности

\*Г. Отчет был составлен по ошибке и удален

**Вопрос 7. Трассировочные таблицы**

Пункт 1 Для чего в проектной документации используются трассировочные таблицы?

\*А. Для упрощения навигации между разделами документов

\*Б. Для упрощения процесса сбора покрытия требований

В. Для представления тестовых примеров в табличном виде

Г. Для определения типов связываемой документации

Пункт 2. Выберите возможные типы ссылок между разделами документов

\*А. Гиперссылка

\*Б. Ссылка требований верхнего уровня на требования нижнего уровня

В. Ссылка между успешно выполненными тестами

\*Г. Ссылка между различными вариантами реализации требований

Пункт 3. Какие элементы используются для организации системы ссылок между документами?

\*А. Якоря

Б. Вехи

\*В. Ссылки

Г. Возвраты

**Вопрос 8. Трассировочные таблицы**

Пункт 1**.** Выберите верные характеристики якоря

\*А. Имеет уникальный идентификатор

Б. Имеет тип

В. Привязан к разделу документа

\*Г. На него может ссылаться несколько ссылок

Пункт 2**.** Выберите верные характеристики ссылки

А. Имеет уникальный идентификатор

\*Б. Имеет тип

В. Привязан к разделу документа

Г. Может ссылаться на несколько якорей

Пункт 3**.** Выберите верные характеристики трассировочной таблицы

\*А. Содержит информацию о типах ссылок

Б. Содержит информацию о типах якорей

\*В. Содержит идентификаторы ссылок

\*Г. Содержит идентификаторы якорей

**Раздел. «Формальные инспекции»**

**Вопрос 1. Процесс формальной инспекции**

Пункт 1 В чем отличия формальной инспекции от обычного обсуждения артефактов проекта?

\*А. Четко определены этапы процесса формальной инспекции

\*Б.В результате формальной инспекции создаются документы, по которым можно судить о замечаниях и проблемах, которые имели место быть в коде

В. Формальная инспекция может применяться только к проектной документации

Г. Формальная инспекция может проводиться как в режиме личной встречи, так и при переписке.

Пункт 2. Укажите обязательных участников формальной инспекции

\*А. автор

Б. представитель службы качества

В. руководитель проекта

\*Г. эксперт

Пункт 3. В чем отличия формальной инспекции от тестирования?

\*А. Не происходит выполнения программного кода

Б. Может применяться как к требованиям, так и к коду

В. Более быстрый процесс

Г. Не фиксируются найденные проблемы

**Вопрос 2. Этапы формальной инспекции**

Пункт 1. Какие роли участников формальной инспекции могут быть совмещены?

\*А. ведущий+эксперт

Б. автор+ведущий

В. автор+эксперт

Г.автор+представитель службы качества

Пункт 2. Какие участники формальной инспекции участвуют в процессе подготовки к инспекции?

А. ведущий

\*Б. инспектор

\*В. автор

Г. руководитель проекта

Пункт 3. Какие участники формальной инспекции участвуют в процессе обсуждения?

\*А. ведущий

\*Б. инспектор

\*В. автор

Г. руководитель проекта

**Вопрос 3. Этапы формальной инспекции**

Пункт 1. Когда можно начинать подготовку к формальной инспекции?

\*А. Инспектируемый документ готов и защищен от изменений на время инспекции

Б. Автор лично присутствует

\*В. Назначен ведущий инспекции

Г. Назначены инспектора

Пункт 2. Когда можно начинать собрание формальной инспекции?

А. Автор лично присутствует

Б. У инспекторов есть замечания к инспектируемому документу

\*В. Ведущий лично пристутствует

\*Г. Хотя бы один инспектор присутствует

Пункт 3. Что можно считать критерием окончания формальной инспекции?

\*А. Инспектируемый документ утвержден или отправлен на доработку

Б. У инспекторов нет замечаний к инспектируемому документу

В. Автор не возражает против замечаний инспекторов

Г. Автор не явился на собрание

**Вопрос 4. Обязанности участников формальной инспекции**

Пункт 1. Укажите обязанности ведущего

\*А. Разослать приглашения участникам инспекции

Б. Отвечать на замечания

В. Выявлять проблемы в инспектируемом документе

\*Г. Проводить собрание

Пункт 2. Укажите обязанности автора

А. Разослать приглашения участникам инспекции

\*Б. Отвечать на замечания

В. Выявлять проблемы в инспектируемом документе

Г. Проводить собрание

Пункт 3. Укажите обязанности инспектора

А. Разослать приглашения участникам инспекции

Б. Отвечать на замечания

\*В. Выявлять проблемы в инспектируемом документе

Г. Проводить собрание

**Вопрос 5. Обязанности участников формальной инспекции**

Пункт 1. В каких случаях формальная инспекция может быть прервана?

\*А. Если при повторной инспекции найдены изменения в инспектируемом объекте, не связанные с замечаниями, указанными на предыдущей инспекции

Б. В случае неявки автора

\*В. В случае, если объем объекта инспекции слишком велик и требуется инспекция в полной форме

Г. В случае, если дискуссия во время собрания непродуктивна

Пункт 2. В каких случаях объект формальной инспекции может быть принят?

\*А. В случае полного отсутствия замечаний от инспекторов

\*Б. В случае, если замечания инспекторов не существенны и не снижают качества инспектируемого документа

\*В. В случае, если замечания инспекторов были сняты во время собрания

Г. В случае, если количество замечаний инспекторов не превышает двух

Пункт 3. В каких случаях назначается повторная инспекция документа?

\*А. В случае, если инспекторами были сформулированы замечания к документу

\*Б. В случае, если автор согласился с замечаниями инспекторов

В. В случае, если на проведении повторной инспекции настаивает ведущий

Г. В случае, если время проведения первого собрания слишком велико

**Вопрос 6. Цели этапов формальной инспекции**

Пункт 1. Укажите цели этапа планирования

\*А. Определить конкретных участников инспекции

\*Б. Определить сроки проведения инспекции

В. Выбрать объект инспекции

Г. Выбрать ведущего

Пункт 2. Укажите цели этапа подготовки

А. Выбрать инспекторов

\*Б. Изучить инспектируемый документ

\*В. Сформулировать замечания к инспектируемому документу

Г. Назначить время обсуждения

Пункт 3. Укажите цели этапа обсуждения

\*А. Довести замечания до сведения автора

Б. Убедиться, что ранее сформулированные замечания исправлены

\*В. Заполнить бланк инспекции

\*Г. Принять решение о принятии или переделке документа

**Вопрос 7. Бланк формальной инспекции**

Пункт 1. Какую информацию заносит ведущий в бланк инспекции на фазе планирования?

\*А. Название объекта инспекции

\*Б. Фамилии автора и инспекторов

\*В. Дату проведения собрания

Г. Решение о принятии или переделке документа

Пункт 2. Какую информацию заносит ведущий в бланк инспекции на фазе собрания?

\*А. Фамилии присутствующих

\*Б. Дату проведения собрания

В. Название объекта инспекции

\*Г. Ответы на контрольные вопросы

Пункт 3. Какую информацию заносит ведущий в бланк инспекции на фазе завершения?

\*А. Решение о принятии или переделке документа

\*Б. Длительность инспекции

В. Фамилии присутствующих

\*Г. Решение о форме проведения повторной инспекции

**Вопрос 8. Контрольные вопросы бланка инспекции**

Пункт 1. Кто отвечает на контрольные вопросы бланка инспекции?

А. Автор

\*Б. Инспектор

\*В. Ведущий

Г. Руководитель проекта

Пункт 2. В каких случаях снимаются замечания по контрольным вопросам бланка инспекции?

\*А. Замечание неприменимо в данном случае

Б. Автор не согласен с замечанием

\*В. Автором, ведущим и инспекторами принято совместное решение о том, что замечание может быть снято

Г. Ведущий настаивает на снятии замечания

Пункт 3. В каких случаях необходимо записывать замечания к объекту инспекции?

\*А. Если на контрольный вопрос получен ответ “Yes”

Б. Если на контрольный вопрос получен ответ “No”

В. Если на контрольный вопрос получен ответ “N/A”

Г. Если все инспекторы сформулировали одинаковые замечание

**Раздел. «Документирование процесса формальной инспекции»**

**Вопрос 1. Документирование процесса формальной инспекции**

Пункт 1 Какими документами может регламентироваться процесс формальной инспекции?

\*А. Стандарт проекта «Формальные инспекции»

\*Б. Стандарт предприятия «Формальные инспекции»

\*В. Часть ю стандарта «Верификация программного обеспечения»

Г. Частью стандарта «Разработка тестов»

Пункт 2. Какие аспекты должны быть освещены в стандарте проекта «Формальные инспекции»?

\*А. Должны быть определены фазы формальной инспекции

\*Б. Должны быть определены роли участников формальной инспекции

\*В. Должны быть определены критерии, разрешающие переход к следующей фазе инспекции

\*Г. Должны быть определены критерии начала формальной инспекции

Пункт 3. Какие аспекты должны быть освещены в стандарте проекта «Формальные инспекции»?

\*А. Формат бланка инспекции

\*Б. Типы инспектируемых документов

\*В. Количество инспекторов

\*Г. Критерий для проведения повторной инспекции

**Вопрос 2. Бланк формальной инспекции**

Пункт 1. Из каких элементов должен состоять бланк формальной инспекции?

\*А. Титульный лист с идентификатором инспекции, инспектируемого документа, принятого решения и т.п.

\*Б. Список контрольных вопросов

\*В. Список несоответствий

Г. Список принятых решений

Пункт 2. Какую информацию помещается на титульный лист бланка формальной инспекции?

\*А. Название и версия объекта инспекции

\*Б. Фамилии ведущего и инспекторов

В. Количество планируемых повторных инспекций

\*Г. Принятое решение

Пункт 3. Какую информацию помещается на титульный лист бланка формальной инспекции?

\*А. Дата начала инспекции

\*Б. Фамилия автора

\*В. Названия и номера версий документов, регламентирующих инспекцию

Г. Идентификаторы запросов на изменение, создаваемых по результатам инспекции

**Вопрос 3. Бланк формальной инспекции**

Пункт 1. Что записывается в списке контрольных вопросов формальной инспекции?

А. Вопросы, обязательные для проверки по мнению инспекторов

\*Б. Вопросы, обязательные для проверки в соответствии со стандартами проекта

\*В. Ответ на вопрос

\*Г. Ссылка на описание несоответствия

Пункт 2. Что записывается в списке несоответствий?

\*А. Описание сути несоответствия

Б. Описание способов устранения несоответствия

\*В. Ссылки на запросы на изменение

\*Г. Отметка об исправлении несоответствия с момента предыдущей инспекции

Пункт 3. Кто вносит информацию о несоответствиях в список на бланке инспекции?

А. Автор

\*Б. Инспектор

\*В. Ведущий

Г. Руководитель проекта

**Вопрос 4. Состояния документов в ходе формальной инспекции**

Пункт 1. В каких состояниях находится документ во время формальной инспекции?

А. Черновик

\*Б. Готов

\*В. Инспектируется

\*Г. Принят

Пункт 2. В каких состояниях допустимо изменение инспектируемого документа?

\*А. Переработка

Б. Готов

В. Инспектируется

Г. Принят

Пункт 3. В каких состояниях документ может находиться на фазе собрания?

А. Готов

Б. Черновик

\*В. Инспектируется

Г. Принят

**Вопрос 5. Состояния документов в ходе формальной инспекции**

Пункт 1. Кто переводит инспектируемый документ из состояния «Переработка» в «Готов»?

\*А. Автор

Б. Инспектор

В. Ведущий

Г. Руководитель проекта

Пункт 2. Кто переводи инспектируемый документ из состояния «Готов» в состояние «Инспектируется»?

А. Автор

Б. Инспектор

\*В. Ведущий

Г. Руководитель проекта

Пункт 3. Кто переводит инспектируемый документ из состояния «Инспектируется» в состояние «Принят»?

А. Автор

Б. Инспектор

\*В. Ведущий

Г. Руководитель проекта

**Вопрос 6. Формальные инспекции программного кода**

Пункт 1. В каких случаях прибегают к формальной инспекции программного кода?

\*А. Когда необходимо исключить наличие недекларированных требованиями участков кода

\*Б. Когда необходимо проверить стиль кодирования

В. В качестве замены регрессионного тестирования

\*Г. Для проверки участков кода, тестирование которых нельзя автоматизировать

Пункт 2. Какие особенности необходимо учитывать при инспекции программного кода

\*А. Выделение памяти

\*Б. Реализация участков кода требованиями

\*В. Робастность кода

Г. Полноту тестирования

Пункт 3. Каким образом можно проверить покрытие программного кода требованиями при формальной инспекции?

\*А. При помощи создания ручной трассировки требования на код

Б. При помощи таблиц сбора покрытия

В. При помощи создания тестовых примеров

Г. При помощи инструментов сбора покрытия

**Вопрос 7. Формальные инспекции проектной документации**

Пункт 1. Какие виды проектной документации могут быть подвергнуты формальной инспекции?

\*А. Системные требования

\*Б. Тест-требования

\*В. Тест-планы

\*Г. Стандарт проекта «Формальные инспекции»

Пункт 2. Что проверяется в ходе формальной инспекции требований?

\*А. Полнота требований

Б. Замкнутость требований

\*В. Непротиворечивость требований

\*Г. Тестопригодность требований

Пункт 3. Какие запросы на изменение могут создаваться по результатам формальной инспекции функциональных требований?

\*А. Запросы на изменение функциональных требований

\*Б. Запросы на изменение кода

\*В. Запросы на изменение системных требований

\*Г. Запросы на изменение стратегии тестирования

**Вопрос 8. Список несоответствий в ходе формальной инспекции**

Пункт 1. Какие несоответствия могут выявляться в ходе формальной инспекции тест-плана?

\*А. Тестовые примеры не соответствуют тест-требованиям

\*Б. Тестовые примеры недостаточно полно покрывают тест-требования

В. Результаты выполнения тестовых сценариев различаются на разных конфигурациях тестовых стендов

\*Г. Тестовые примеры противоречат друг другу

Пункт 2. Какие несоответствия могут выявляться в ходе формальной инспекции отчета о покрытии?

\*А. Тесты недостаточно покрывают код

\*Б. Выбран неверный тип покрытия

В. Тесты противоречат друг другу

\*Г. Отчет о покрытии имеет неверный формат

Пункт 3. Какие несоответствия могут выявляться в ходе формальной инспекции тест-требований?

\*А. Тест-требования противоречат функциональным требованиям

\*Б. Тест-требования невозможно протестировать

\*В. Тест-требования дублируют друг друга

Г. Тест-требования будут удалены в последующих версиях программной системы

**Раздел. «Модульное тестирование»**

**Вопрос 1. Модульное тестирование**

Пункт 1 Каковы цели проведения модульного тестирования?

\*А. Локализовать дефекты на ранних стадиях тестирования

\*Б.Упростить интеграционное тестирование

В. Ограничить тестирование системы только тестированием модулей

Г. Ускорить регрессионное тестирование

Пункт 2 Какие задачи решаются в ходе модульного тестирования?

\*А. Поиск и документирование дефектов

\*Б.Поддержка рефакторинга модулей

В. Отладка модулей

\*Г. Уточнение межмодульных интерфейсов

Пункт 3 Какие составные части обычно входят в тестовое окружене для модульного тестирования?

\*А. Драйвер

\*Б. Заглушки

В. Тест-требования

Г. Описание конечных автоматов

**Вопрос 2. Границы модулей**

Пункт 1 Что можно считать модулем, подвергаемым тестированию?

\*А. Модуль – минимальная единица компиляции системы

\*Б.Модуль – набор классов с единым интерфейсом

В. Модуль – набор различных участков программного кода, выполняющих разные функции

\*Г. Модуль – это часть программного кода, реализующая группу требований

Пункт 2 Каким образом проводится тестирование модулей?

\*А. Драйвер вызывает функции тестируемого модуля, заглушки имитируют поведение остальных частей системы

Б.Драйвер вызывает функции тестируемого модуля, заглушки отсуствуют

\*В. Драйвер вызывает функции тестируемого модуля, модуль работает совместно с остальными частями системы

Г. Драйвер вызывает функции тестируемого модуля, заглушки представляют собой пустые функции

Пункт 3 Что не может изменяться в процессе тестирования модуля до создания соответствующих запросов на изменение?

\*А. Ррограммный код тестируемого модуля

Б. Программный код заглушек

В. Программный код драйвера

Г. Внешние интерфейсы модуля

**Вопрос 3. Организация модульного тестирования**

Пункт 1. Какие задачи решаются в ходе планирования модульного тестирования?

\*А. Определяются требования к ресурсам, необходимым для тестирования

\*Б. Определяются подходы к тестированию

\*В. Определяется план-график работ

Г. Определяется точное количество тест-требований

Пункт 2.Какие задачи решаются в ходе определения свойств системы, подлежащей тестированию?

\*А. Изучаются функциональные требования

\*Б. Определяются дополнительные требования

В. Пишутся тестовые примеры

\*Г. Определяются классы эквивалентности входных и выходных данных

Пункт 3. Какие задачи решаются в ходе разработки тестов для модульного тестирования?

А. Разрабатывается общая архитектура тестового набора

Б. Разрабатываются тестовые примеры

В. Составляются тест-требования

Г. Выполняются тесты

**Вопрос 4. Организация модульного тестирования**

Пункт 1. Какие задачи решаются в ходе оценки проведенного тестирования?

\*А. Отмечаются отличия реально проведенного тестирования от запланированного

\*Б. Определяются отличия поведения модуля от заданного в требованиях

\*В. Составляется отчет о покрытии

Г. Определяется критерий полноты покрытия

Пункт 2. Какая информация собирается по результатам этапа выполнения тестов?

\*А. Результат выполнения каждого тестового примера

\*Б. Информация о покрытии

В. Информация о типе тестирования

\*Г. Информация о конфигурации тестового стенда

Пункт 3. Что создается на этапе реализации тестов?

А. Тест-план

\*Б. Тестовое окружение

\*В. Тестовые наборы данных

\*Г. Формальное описание тестовых примеров

**Вопрос 5. Тестирование классов**

Пункт 1. Какие из перечисленных ниже дефектов можно отнести к дефектам инкапсуляции?

А. При создании объекта класса начальные значения его свойств не соответствуют требованиям

Б. При наследовании классов оказываются недоступными некоторые методы

\*В. При помощи методов класса доступны не весь диапазон значений свойств класса

Г. Классы-наследники не имеют доступа к данным родителя

Пункт 2. Какие из перечисленных ниже дефектов можно отнести к дефектам инстанцирования?

\*А. При создании объекта класса начальные значения его свойств не соответствуют требованиям

Б. При наследовании классов оказываются недоступными некоторые методы

В. При помощи методов класса доступны не весь диапазон значений свойств класса

Г. Классы-наследники не имеют доступа к данным родителя

Пункт 3. Какие из перечисленных ниже дефектов можно отнести к дефектам наследования?

А. При создании объекта класса начальные значения его свойств не соответствуют требованиям

\*Б. При наследовании классов оказываются недоступными некоторые методы

В. При помощи методов класса доступны не весь диапазон значений свойств класса

\*Г. Классы-потомки не имеют доступа к данным родителя

**Вопрос 6. Тестирование классов**

Пункт 1. Какие подходы могут использоваться для определения полноты тестирования класса?

\*А. Выполнены все строки программного кода методов класса

\*Б. Выполнены все возможные сценарии работы методов класса

В. Выполнена проверка согласованности интерфейсов класса

\*Г. Выполнена проверка корректного вызова всех методов класса

Пункт 2. Какие проблемы при тестировании классов возникают при изменении внутреннего состояния объекта класса?

\*А. Сценарии работы методов объекта могут меняться в различных его состояниях

Б. Интерфейс методов может меняться в различных состояниях объекта

В. Наследование класса может меняться в различных состояниях объекта

Г. Инкапсуляция данных может меняться в различных состояниях объекта

Пункт 3. Какие проблемы могут быть выявлены в результате тестирования изменений класса?

\*А. Несовпадение интерфейсов с остальными классами

\*Б. Появление новых внутренних состояний

\*В. Проблемы наследования

Г. Проблемы компиляции

**Вопрос 7. Особенности подходов к разработке тестового окружения для модульного тестирования**

Пункт 1. Какие особенности имеет подход к разработке тестового окружения при модульном тестировании с использованием заглушек?

\*А. Повышенная трудоемкость разработки тестового окружения за счет повышенной сложности драйвера

\*Б. Повышенная трудоемкость разработки тестового окружения за счет наличия заглушек

В. Простота отладки тестового окружения

\*Г. Более простая локализация дефектов в тестируемом модуле

Пункт 2. Какие особенности имеет подход к разработке тестового окружения при модульном тестировании без использования заглушек?

\*А. Сокращение трудоемкости разработки драйвера

Б. Сокращение трудоемкости написания тестовых примеров

\*В. Маскировка дефектов из-за совместной работы модулей

\*Г. Увеличение трудоемкости написания тестовых сценариев

Пункт 3. Какие функции выполняют драйвер и заглушки при модульном тестировании?

\*А. Моделируют внутреннее состояние системы, использующей тестируемый модуль

\*Б. Моделируют поведение системы в различных состояниях

\*В. Создают тестовое окружение

Г. Выявляют дефекты тестируемого модуля

**Вопрос 8. Риски модульного тестирования**

Пункт 1. Какие риски существуют на этапе планирования модульного тестирования?

\*А. Недооценено количество ресурсов, необходимых для тестирования

\*Б. Неверно определены подходы к тестированию

\*В. План-график работ слишком оптимистичен

Г. Неверно определено точное количество тест-требований

Пункт 2.Какие риски существуют при определении свойств системы, подлежащей тестированию?

\*А. Недостаточная полнота тест-требований, ведущая к невозможности создания полного тест-плана

\*Б. Наличие неучтенных дополнительных требований

В. Отсутствуют тест-планы с предыдущих этапов проекта

\*Г. Невозможно однозначно определить классы эквивалентности входных данных системы на основе существующих требований

Пункт 3. Какие риски существуют в ходе разработки тестов для модульного тестирования?

\*А. Общая архитектура тестового набора непригодна для дальнейшего расширения

\*Б. Тестовые примеры разрабатываются слишком медленно

\*В. Разработанный набор тестов не является полным

Г. Выполнение тестов задерживается из-за неготовности тестового стенда

**Раздел. «Интеграционное тестирование»**

**Вопрос 1. Интеграционное тестирование**

Пункт 1 Каковы цели интеграционного тестирования?

\*А. Проверить полноту взаимодействия модулей системы

\*Б. Проверить корректность межмодульных интерфейсов

В. Проверить возможность совместной сборки модулей

Г. Проверить внутреннюю непротиворечивость отдельных модулей

Пункт 2. Какие задачи решаются в ходе интеграционного тестирования?

\*А. Совместная сборка тестируемых модулей и тестового окружения

\*Б. Выполнение интеграционных тестовых сценариев

\*В. Сбор несоответствий интерфейсов модулей

Г. Устранение недостатков интерфейсов модулей

Пункт 3. Какие изменения могут вноситься в систему по результатам интеграционного тестирования?

\*А. Изменения в отдельные модули

\*Б. Изменения в интерфейсы межмодульного взаимодействия

\*В. Изменения в требования

\*Г. Изменения в архитектуру

**Вопрос 2. Интеграционное тестирование**

Пункт 1. Какие типы интеграционного тестирования вы знаете?

\*А. Восходящее тестирование

Б. Компонентное тестирование

\*В. Нисходящее тестирование

\*Г. Монолитное тестирование

Пункт 2. Чем отличаются друг от друга различные типы интеграционного тестирования?

\*А. Последовательностью интеграции модулей

\*Б. Частотой интеграции модулей

В. Размером интегрируемых модулей

\*Г. Количеством модулей, интегрируемых на каждом этапе

Пункт 3. Какие дефекты могут быть выявлены в ходе интеграционного тестирования?

\*А. Дефекты межмодульных интерфейсов

\*Б. Дефекты внутренней логики отдельных модулей

В. Дефекты сборки модулей

\*Г. Дефекты логики совместной работы модулей

**Вопрос 3. Методы интеграционного тестирования**

Пункт 1. Укажите основные достоинства восходящего интеграционного тестирования

\*А. Упрощается локализация дефектов в модулях

Б. Упрощается процедура разработки тестового окружения

В. Уменьшается количество работ по сборке модулей

\*Г. Дефекты отдельных модулей выявляются на ранних стадиях тестирования

Пункт 2. Укажите основные достоинства нисходящего интеграционного тестирования

А. Упрощается локализация дефектов в модулях

\*Б. Упрощается процедура разработки тестового окружения

В. Уменьшается количество работ по сборке модулей

\*Г. Дефекты отдельных модулей выявляются на ранних стадиях тестирования

Пункт 3. Укажите основные достоинства монолитного интеграционного тестирования

А. Упрощается локализация дефектов в модулях

\*Б. Упрощается процедура разработки тестового окружения

\*В. Уменьшается количество работ по сборке модулей

Г. Дефекты отдельных модулей выявляются на ранних стадиях тестирования

**Вопрос 4. Методы интеграционного тестирования**

Пункт 1. Укажите основные недостатки восходящего интеграционного тестирования

\*А. Необходимо разрабатывать драйвер и заглушки

Б. Дефекты отдельных модулей выявляются на финальных стадиях проекта

В. Затруднена локализация дефектов в модулях

Г. Процесс тестирования плохо автоматизируется

Пункт 2. Укажите основные недостатки нисходящего интеграционного тестирования

\*А. Необходимо разрабатывать заглушки

Б. Дефекты отдельных модулей выявляются на финальных стадиях проекта

В. Затруднена локализация дефектов в модулях

Г. Процесс тестирования плохо автоматизируется

Пункт 3. Укажите основные недостатки монолитного интеграционного тестирования

А. Необходимо разрабатывать драйвер и заглушки

\*Б. Дефекты отдельных модулей выявляются на финальных стадиях проекта

\*В. Затруднена локализация дефектов в модулях

\*Г. Процесс тестирования плохо автоматизируется

**Вопрос 5. Методы интеграционного тестирования**

Пункт 1. Назовите известные вам типы интеграционного тестирования

\*А. Тестирование с поздней интеграцией

\*Б. Тестирование с ранней интеграцией

\*В. Тестирование с регулярной (постоянной) интеграцией

Г. Тестирование с эпизодической интеграцией

Пункт 2. Пусть R — разработка требований на отдельный модуль, C — разработка программного кода, V — тестирование модуля, I — интеграционное тестирование всего, что было сделано раньше. Укажите цепочку, изображающую тестирование с поздней интеграцией.

\*А. R-C-V-R-C-V-R-C-V-I-R-C-V-R-C-V-I

Б. R-C-V-I-R-C-V-I-R-C-V-I

В. R-I-R-I-R-I-R-I

C. C-I-C-I-C-I-C-I-C-I

Пункт 3. Пусть R — разработка требований на отдельный модуль, C — разработка программного кода, V — тестирование модуля, I — интеграционное тестирование всего, что было сделано раньше. Укажите цепочку, изображающую тестирование с постоянной интеграцией.

А. R-C-V-R-C-V-R-C-V-I-R-C-V-R-C-V-I

\*Б. R-C-I-R-C-I-R-C-I

В. R-I-R-I-R-I-R-I

C. C-I-C-I-C-I-C-I-C-I

**Вопрос 6. Методы интеграционного тестирования**

Пункт 1. Укажите основные недостатки тестирования с поздней интеграцией

\*А. Интеграционное тестирование откладывается на поздние стадии проекта

\*Б. Затруднена локализация дефектов межмодульных интерфейсов

В. Нужна разработка драйверов и заглушек

Г. Велики трудозатраты на сборку тестируемых групп модулей

Пункт 2. Укажите основные недостатки тестирования с ранней интеграцией

А. Интеграционное тестирование откладывается на поздние стадии проекта

Б. Затруднена локализация дефектов межмодульных интерфейсов

\*В. Нужна разработка драйверов и заглушек

\*Г. Велики трудозатраты на сборку тестируемых групп модулей

Пункт 3. Укажите основные недостатки тестирования с постоянной интеграцией

А. Интеграционное тестирование откладывается на поздние стадии проекта

Б. Затруднена локализация дефектов межмодульных интерфейсов

\*В. Нужна разработка заглушек

\*Г. Велики трудозатраты на сборку тестируемых групп модулей

**Вопрос 7. Методы интеграционного тестирования**

Пункт 1. Укажите основные достоинства тестирования с поздней интеграцией

\*А. Подходит для тестирования слабо связанных систем

Б. Не требуется разработка заглушек

В. Не требуется разработка драйвера

Г. Быстро выявляются проблемы межмодульных интерфейсов

Пункт 2. Укажите основные достоинства тестирования с ранней интеграцией

А. Подходит для тестирования слабо связанных систем

Б. Не требуется разработка заглушек

В. Не требуется разработка драйвера

\*Г. Быстро выявляются проблемы межмодульных интерфейсов

Пункт 3. Укажите основные достоинства тестирования с постоянной интеграцией

А. Подходит для тестирования слабо связанных систем

\*Б. Не требуется разработка заглушек

В. Не требуется разработка драйвера

\*Г. Быстро выявляются проблемы межмодульных интерфейсов

**Вопрос 8. Планирование интеграционного тестирования**

Пункт 1. Какие документы определяют последовательность интеграции модулей при проведении интеграционного тестирования?

\*А. Стратегия тестирования и интеграции

Б. Интеграционный тест-план

В. Интеграционные тест-требования

\*Г. Интеграционная архитектура

Пункт 2. Укажите основные способы планирования последовательности интеграционного тестирования при кластерном подходе

\*А. Для каждого кластера определяются кластеры, которые зависят от него

\*Б. Для каждого кластера определяются кластеры, которые должны быть протестированы до него

В. Для каждого кластера создается отдельный тест-план

Г. Для каждого кластера определяется состав модулей

Пункт 3. Укажите основные риски и причины возникновения рисков процесса интеграционного тестирования

\*А. Не предусмотренная несогласованность интерфейсов тестируемых модулей

\*Б. Несоблюдение план-графика разработки и тестирования отдельных модулей

В. Несоблюдение план-графика приемо-сдаточных испытаний

\*Г. Непригодность выбранного метода интеграции для выявления всех дефектов

**Раздел. «Комплексное (системное) тестирование»**

**Вопрос 1. Виды системного тестирования**

Пункт 1 Укажите известные вам виды системного тестирования

\*А. Функциональное тестирование

Б. Структурное тестирование

\*В. Тестирование безопасности

Г. Восходящее тестирование

Пункт 2 Укажите известные вам виды системного тестирования

\*А. Тестирование производительности

\*Б. Стрессовое тестирование

В. Тестирование мемодульных интерфейсов

Г. Тестирование документации

Пункт 2 Укажите известные вам виды системного тестирования

\*А. Тестирование конфигурации

\*Б. Тестирование удобства использования

В. Тестирование модулей

Г. Нисходящее тестирование

**Вопрос 2. Функциональное системное тестирование**

Пункт 1. Выберите верные утверждения

\*А. Функциональное системное тестирование предназначено для проверки того, что система ведет себя в соответствии с ожиданиями пользователя (выраженными в виде системных требований)

Б. Функциональное системное тестирование проводится методом стеклянного ящика

В. Для функционального системного тестирования невозможно определить критерии полноты тестирования

\*Г. Одна из основных сложностей системного тестирования – локализация дефектов

Пункт 2. Какие критерии покрытия используются при функциональном системном тестировании?

\*А. Покрытие требований

Б. Покрытие строк кода

В. Покрытие условий

\*Г. Покрытие классов эквивалентности входных и выходных данных

Пункт 3. Какие классы входных и выходных данных должны быть покрыты при функциональном системном тестировании?

\*А. Все классы допустимых входных данных должны корректно обрабатываться системой

\*Б. Все классы недопустимых выходных данных должны корректно отбрасываться системой

\*В. Во время тестирования система должна побывать во всех своих внутренних состояниях

\*Г. Во время тестирования должны генерироваться все возможные классы выходных данных системы

**Вопрос 3. Тестирование производительности**

Пункт 1. Укажите основные цели тестирования производительности

А. Определение того, что система корректно работает на любой конфигурации оборудования

\*Б. Определение того, что время отклика системы соответствует указанному в требованиях при любом количестве пользовательских запросов в единицу времени

\*В. Определение того, что система имеет достаточный (и определенный в требованиях) уровень производительности на любой конфигурации оборудования

Г. Определение того, что система сохраняет возможность стабильной работы при любом количестве запросов пользователей в единицу времени

Пункт 2. Какие данные сохраняют в отчетах о тестировании производительности?

\*А. Загрузка аппаратного и программного обеспечения

\*Б. Скоростные характеристики тестируемой системы

\*В. Конфигурация тестового стенда

Г. Покрытие функциональных требований тестами производительности

Пункт 3. Какие скоростные характеристики тестируемой системы можно определять в ходе тестирования производительности?

\*А. Количество обработанных в единицу времени запросов

Б. Количество сбоев в единицу времени

\*В. Равномерность времени отклика системы в разное время

\*Г. Количество отказов в обслуживании при превышении допустимой нагрузки

**Вопрос 4. Стрессовое тестирование**

Пункт 1. Укажите основные задачи стрессового тестирования

А. Определение того, что система корректно работает на любой конфигурации оборудования

\*Б. Определение того, что система сохраняет работоспособность при при количестве пользовательских запросов, превышающих максимально допустимое значение в единицу времени

В. Определение того, что система имеет достаточный (и определенный в требованиях) уровень производительности на любой конфигурации оборудования

\*Г. Определение того, что система сохраняет возможность стабильной работы при критической нехватке ресурсов

Пункт 2. Укажите основные типы систем, которые нужно обязательно подвергать стрессовому тестированию

А. Исследовательские прототипы программ для научных расчетов

\*Б. Web-системы

\*В. Системы распределенного документооборота

\*Г. Межсетевые экраны

Пункт 3. Укажите основные цели системного тестирования

\*А. Обнаружить условия, при которых нормальная работа системы будет нарушена

Б. Определить характеристики производительности системы при нормальном уровне нагрузки

В. Промоделировать работу системы в нормальных условиях

\*Г. Определить максимально допустимую нагрузку на систему

**Вопрос 5. Тестирование безопасности**

Пункт 1. Для каких систем рекомендуется применять тестирование безопасности?

А. Для систем, хранящих личные данные пользователей

Б. Для систем, хранящих данные, имеющие коммерческую ценность

В. Для систем учебного назначения

\*Г. Для систем передачи секретных данных

Пункт 2. Что проверяется в ходе тестирвоания безопасности?

\*А. Что информация не теряется и не повреждается

\*Б. Что информацию невозможно перехватить или подменить

\*В. Что к информации невозможно получить несанкционированный доступ

Г. Что информация доступна пользователю только средствами одной системы

Пункт 3. Какие группы свойств системы подлежат проверке при тестировании безопасности?

\*А. Механизмы аутентификации и идентификации

\*Б. Механизмы аудита событий

\*В. Гарантии проектирования

Г. Открытость алгоритмов

**Вопрос 6. Сертификация**

Пункт 1. Кто участвует в процессе сертификации?

\*А. Заявитель

\*Б. Сертифицирующий орган

\*В. Наблюдательный орган

Г. Регулирующий орган

Пункт 2. Что проверяется в ходе сертификации?

А. Соответствие реализации системы требованиям на систему

\*Б. Соответствие процессов и артефактов разработки требованиям стандартов

В. Соответствие требований на систему плану сертификации

Г. Соответствие требований сертифицируещего органа к процессу управления конфигурациями проекта

Пункт 3. Укажите типы сертификатов на программное обеспечение

\*А. Сертификат качества

\*Б. Сертификат соответствия

В. Сертификат тестопригодности

Г. Сертификат полноты

**Вопрос 7. Сертификация**

Пункт 1. Что включает в себя план сертификационных испытаний (в соответствии с требованиями стандарта DO-178B)?

\*А. Обзор системы

Б. Отчет о покрытии системы тестами

\*В. Жизненный цикл ПО

Г. Отчет о выполнении тестов

Пункт 2. Что включает в себя план сертификационных испытаний (в соответствии с требованиями стандарта DO-178B)?

\*А. Обзор программного обеспечения

Б. Требования на систему

\*В. Сертификационные соображения

Г.Исходный код системы

Пункт 3. Что включает в себя план сертификационных испытаний (в соответствии с требованиями стандарта DO-178B)?

\*А. Перечень данных жизненного цикла разработки системы

\*Б. План-график работ по сертификации

\*В. Особенности процесса сертификации

Г.Тест-план

**Вопрос 8. Сертификация**

Пункт 1. Что включает в себя итоговое заключение по программному обеспечению (в соответствии с требованиями стандарта DO-178B)?

\*А. Обзор системы

Б. Отчет о покрытии системы тестами

\*В. Жизненный цикл ПО

Г. Отчет о выполнении тестов

Пункт 2. Что включает в себя итоговое заключение по программному обеспечению (в соответствии с требованиями стандарта DO-178B)?

\*А. Обзор программного обеспечения

Б. Требования на систему

\*В. Сертификационные соображения

Г.Исходный код системы

Пункт 3. Что включает в себя итоговое заключение по программному обеспечению (в соответствии с требованиями стандарта DO-178B)?

\*А. Перечень данных жизненного цикла разработки системы

Б. План-график работ по сертификации

\*В. Историю изменений

Г.Тест-план

**Вопрос 9. Типы системного тестирования**

Пункт 1. Укажите цели тестирования конфигурации

А. Проверка того, что система корректно работает со всеми возможными форматами файлов конфигурации

\*Б. Проверка того, что система корректно работает на всех возможных конфигурациях оборудования

\*В. Проверка того, что система корректно работает совместно с другими программными системами

Г. Проверка того, что система корректно работает при любом количестве пользователей

Пункт 2. Укажите цели тестирования надежности и восстановления после сбоев

\*А. Проверка того, что система не теряет данные в результате сбоя

\*Б. Проверка того, что система успешно продолжает работу после сбоя

В. Проверка того, что после сбоя пользователь имеет возможность создать данные заново

Г. Проверка того, что сбой в системе не возникнет никогда

Пункт 3. Укажите цели тестирования удобства использования

А. Проверка того, что системой удобно пользоваться всем ее пользователям

\*Б. Проверка того, что интерфейс системы соответствует заданным критериям удобства

В. Проверка того, что система подстраивается под уровень пользователя

Г. Проверка покрытия функций системы, реализующих пользовательский интерфейс

**Раздел. «Тестирование пользовательского интерфейса»**

**Вопрос 1. Тестирование пользовательского интерфейса**

Пункт 1. Какие аспекты программной системы проверяются при тестировании пользовательского интерфейса?

\*А. Полнота охвата функциональности системы компонентами интерфейса

\*Б. Соответствие интерфейса системы стандартам (общепринятым или локальным), принятым для интерфейсов такого же типа

В. Реализация API систем

\*Г. Степень эффективности человеко-машинного взаимодействия

Пункт 2. Какие аспекты интерфейса программной системы не всегда могут быть определены в качестве четких требований, а только в виде рекомендаций?

А. Полнота охвата функциональности системы компонентами интерфейса

Б. Соответствие интерфейса системы стандартам (общепринятым или локальным), принятым для интерфейсов такого же типа

В. Работа функций системы, вызываемых компонентами пользовательского интерфейса

\*Г. Степень эффективности человеко-машинного взаимодействия

Пункт 3. Какие типы документов создаются для тестирования пользовательского интерфейса?

\*А. Тест-требования

\*Б. Тест-планы

\*В. Методики проверки удобства использования

\*Г. Методики определения полноты покрытия

**Вопрос 2. Функциональное тестирование пользовательских интерфейсов**

Пункт 1. Что, как правило, представляют собой тест-планы для функционального тестирования пользовательских интерфейсов?

\*А. Сценарии на естественном языке, описывающие последовательность действий тестировщика

\*Б. Сценарии на специализированном языке, описывающие последовательность действий программы автоматизации тестирования

В. Сценарии на одном из языков общего назначения для вызова функций системы

Г. Сценарии на на специализированном языке для вызова функций системы

Пункт 2. Что, как правило, анализируется для определения успешного или неуспешного прохождения тестового примера при тестировании пользовательского интерфейса

\*А. Выводимые на экран формы и их элементы

Б. Значения выходных переменных модулей

\*В. Выводимый на экран текст сообщений

Г. Состояние файловой системы

Пункт 3. Какие проблемы могут быть указаны в отчетах о проблемах тестирования пользовательского интерфейса?

\*А. Требование невозможно протестировать

\*Б. Поведение системы не соответствует требованиям

\*В. Пользовательский интерфейс допускает выполнение функций, не определенных в требованиях

\*Г. Пользовательский интер+фейс не соответствует критериям удобства использования

**Вопрос 3. Требования к пользовательским интерфейсам**

Пункт 1. Какие виды требований к пользовательским интерфейсам вы знаете?

\*А. Требования к внешнему виду пользовательского интерфейса

\*Б. Требования к удобству пользовательского интерфейса

В. Требования к покрытию пользовательского интерфейса

\*Г. Требования по доступу к функциям системы при помощи пользовательского интерфейса

Пункт 2. Какие виды требований к внешнему виду пользовательского интерфейса вы знаете?

\*А. Требования к размещению элементов управления на экранных формах

Б. Требования к квалификации пользователя

\*В. Требования к содержанию и оформлению выводимых сообщений

\*Г. Требования к форматам ввода

Пункт 3. Какие виды требований на доступ к внутренней функциональности системы вы знаете?

\*А. Требования к реакции системы на ввод пользователя

Б. Требования к реакции пользователя на вывод системы

В. Требования к форматам вывода

\*Г. Требования к времени отклика системы на ввод пользователя

**Вопрос 4. Полнота покрытия пользовательского интерфейса**

Пункт 1. Что должно достигаться для обеспечения полного функционального покрытия пользовательского интерфейса?

\*А. Все требования к пользовательскому интерфейсу должны быть покрыты

Б. Каждый интерфейсный элемент должен быть использован хотя бы один раз

В. Каждый интерфейсный элемент должен побывать во всех состояниях

Г. Каждый интерфейсный элемент должен побывать во всех состояниях в различных режимах работы системы

Пункт 2. Что должно достигаться для обеспечения полного структурного покрытия пользовательского интерфейса?

А. Все требования к пользовательскому интерфейсу должны быть покрыты

\*Б. Каждый интерфейсный элемент должен быть использован хотя бы один раз

В. Каждый интерфейсный элемент должен побывать во всех состояниях

Г. Каждый интерфейсный элемент должен побывать во всех состояниях в различных режимах работы системы

Пункт 3. Что должно достигаться для обеспечения полного структурного покрытия с учетом состояний элементов пользовательского интерфейса?

А. Все требования к пользовательскому интерфейсу должны быть покрыты

Б. Каждый интерфейсный элемент должен быть использован хотя бы один раз

\*В. Каждый интерфейсный элемент должен побывать во всех состояниях

Г. Каждый интерфейсный элемент должен побывать во всех состояниях в различных режимах работы системы

**Вопрос 5. Ручное тестирование пользовательских интерфейсов**

Пункт 1. Проведение какого вида тестирования затруднено при ручном тестировании интерфейсов?

А. Модульного

\*Б. Регрессионного

В. Интеграционного

Г. Удобства использования

Пункт 2. Укажите основные достоинства ручного тестирования

\*А. Снижаются затраты на разработку тестового окружения

\*Б. Восприятие системы максимально близко к восприятию пользователем

В. Тестовые сценарии легко использовать при регрессионном тестировании

\*Г. Корректные тестовые сценарии однозначно воспринимаются любым тестировщиком

Пункт 3. Укажите основные недостатки ручного тестирования

\*А. Тестовые сценарии на естественном языке могут быть недоступны тестировщикам, говорящим на другом языке

Б. Восприятие системы неоднозначно и зависит от тестировщика

\*В. Тестовые сценарии сложно использовать при регрессионном тестировании

Г. Отсутствуют критерии успешного или неуспешного прохождения теста

**Вопрос 6. Автоматизированное тестирование пользовательских интерфейсов**

Пункт 1. Проведение какого вида тестирования затруднено при автоматизированном тестировании интерфейсов?

А. Модульного

Б. Регрессионного

В. Интеграционного

\*Г. Удобства использования

Пункт 2. Укажите основные достоинства автоматизированного тестирования

А. Снижаются затраты на разработку тестового окружения

Б. Восприятие системы максимально близко к восприятию пользователем

\*В. Тестовые сценарии легко использовать при регрессионном тестировании

\*Г. Корректные тестовые сценарии однозначно воспринимаются любым тестировщиком

Пункт 3. Укажите основные недостатки автоматизированного тестирования

\*А. Тестовые сценарии зависят от точного расположения элементов управления на экране

Б. Восприятие системы неоднозначно и зависит от тестировщика

В. Тестовые сценарии сложно использовать при регрессионном тестировании

\*Г. Требуется модификация большого числа сценариев, если измененился интерфейсный элемент, используемый в начале этих сценариев

**Вопрос 7. Тестирование удобства использования пользовательских интерфейсов**

Пункт 1. Дайте определение удобства использования пользовательского интерфейса

\*А. Это показатель качества интерфейса, определяющий количество усилий, необходимых для изучения принципов работы с программной системой при помощи данного интерфейса, ее использования, подготовки входных данных и интерпретации выходных

Б. Это степень удовлетворенности пользователей при работе с интерфейсом

\*В. Это степень простоты использования интерфейса

Г. Это функциональная полнота интерфейса

Пункт 2. Какие факторы влияют на удобство пользовательского интерфейса?

\*А. Общая удовлетворенность пользователя

\*Б. Простота и эффективность обучения

В. Количество доступных функций

\*Г. Количество ошибок при работе

Пункт 3. Какие виды тестирования пользовательского интерфейса вы знаете?

\*А. Исследовательское

\*Б. Валидационное

\*В. Сравнительное

\*Г. Оценочное

**Вопрос 8. Эвристические проверки удобства использования пользовательских интерфейсов**

Пункт 1. Какие эвристические проверки должны выполняться при проверке удобства использования пользовательских интерфейсов?

\*А. Система должна оповещать пользователя о том, что она делает в каждый момент времени

\*Б. Система должна использовать терминологию предметной области пользователя

В. Система должна корректно использовать средства операционной системы

\*Г. Документация на систему должна быть структурирована для удобства пользователя, а не разработчика

Пункт 2. Какие эвристические проверки должны выполняться при проверке удобства использования пользовательских интерфейсов?

\*А. В интерфейсе должны быть предусмотрены горячие клавиши или иные средства быстрого доступа к функциям

Б. Интерфейс системы не должен отличаться от интерфейса операционной системы

\*В. Интерфейс должен предотвращать ошибки пользователя

\*Г. Интерфейс должен быть стандартизован

Пункт 3. Какие эвристические проверки должны выполняться при проверке удобства использования пользовательских интерфейсов?

\*А. Сообщения об ошибках должны быть понятны пользователю

Б. Сообщения об ошибках должны содержать максимум технической информации

\*В. Использование интерфейса должно быть логичным, а не опираться на запоминание пользователем действий

\*Г. Интерфейс не должен содержать неиспользуемые и мешающие элементы

**Раздел. «Методы разработки устойчивого кода»**

**Вопрос 1. Дефекты в программных системах**

Пункт 1. Какие проявления может иметь дефект в программной системе?

\*А. Сбой

Б. Перебой

\*В. Отказ

\*Г. Авария

Пункт 2. Что является причиной отказных ситуаций?

\*А. Сбой

Б. Перебой

\*В. Отказ

Г. Авария

Пункт 3. Что является причиной аварийных ситуаций?

А. Сбой

Б. Отказ

\*В. Авария

Г. Катастрофа

**Вопрос 2. Дефекты в программных системах**

Пункт 1. Дайте определение сбоя

\*А. Проявление дефекта, которое имеет небольшую продолжительность во времени, может быть устранено без длительных процедур восстановления, и которое не нарушает работу системы в целом

Б. Проявление дефекта, при котором система или ее часть выходят из строя, переставая обслуживать запросы пользователей

В. Проявление дефекта, при котором система или ее часть выходит из строя таким образом, что восстановление нормальной ее работы невозможно или занимает длительное время

Г. Проявление дефекта, при котором выход системы из строя вызывает полный выход системы из строя без возможности восстановления

Пункт 2. Дайте определение отказа

А. Проявление дефекта, которое имеет небольшую продолжительность во времени, может быть устранено без длительных процедур восстановления, и которое не нарушает работу системы в целом

\*Б. Проявление дефекта, при котором система или ее часть выходят из строя, переставая обслуживать запросы пользователей

В. Проявление дефекта, при котором система или ее часть выходит из строя таким образом, что восстановление нормальной ее работы невозможно или занимает длительное время

Г. Проявление дефекта, при котором выход системы из строя вызывает полный выход системы из строя без возможности восстановления

Пункт 3. Дайте определение аварии

А. Проявление дефекта, которое имеет небольшую продолжительность во времени, может быть устранено без длительных процедур восстановления, и которое не нарушает работу системы в целом

Б. Проявление дефекта, при котором система или ее часть выходят из строя, переставая обслуживать запросы пользователей

\*В. Проявление дефекта, при котором система или ее часть выходит из строя таким образом, что восстановление нормальной ее работы занимает очень длительное время

\*Г. Проявление дефекта, при котором выход системы из строя вызывает полный выход системы из строя без возможности восстановления

**Вопрос 3. Сбои в программном обеспечении**

Пункт 1. Какие виды сбоев в программном обеспечении могут вызывать сбойные и отказные ситуации?

\*А. Сбои в системном ПО

\*Б. Сбои в приложении

В. Сбои тестирования

\*Г. Сбои-следствие неверной технологии использования

Пункт 2. Какая информация может помочь локализовать дефект, вызвавший сбой?

\*А. Точка возникновения сбоя

\*Б. Информационное окружение системы в момент сбоя

\*В. Сообщение о сбое

\*Г. Запрос на изменение по результатам сбоя

Пункт 3. В каких случах можно говорить, что произошел сбой?

\*А. Работоспособность системы нарушена на небольшое время

Б. Произошла полная потеря данных

В. Система прекратила выполнение на длительный срок

\*Г. Произошла порча небольшого количества данных

**Вопрос 4. Классификация отказов**

Пункт 1. Как классифицируются отказы по ГОСТ 27.002-89?

\*А. По временным характеристикам

\*Б. По способу обнаружения

\*В. По причинам

\*Г. По связи с другими отказами

Пункт 2. Какие виды отказов относятся к классификации по причинам отказов согласно ГОСТ 27.002-89?

А. Деградационный отказ

\*Б. Ресурсный отказ

В. Скрытый отказ

\*Г. Производственный отказ

Пункт 3. Какие виды отказов относятся к классификации по временным характеристикам отказов согласно ГОСТ 27.002-89?

\*А. Деградационный отказ

\*Б. Постепенный отказ

В. Эксплуатационный отказ

Г. Явный отказ

**Вопрос 5. Защитное программирование**

Пункт 1. Дайте определение защитного программирования

А. Это метод разработки программных систем, при котором тестирование становится не нужным

\*Б. Это метод разработки программных систем, при котором сбои не приводят к отказам и авариям

В. Это метод разработки программных систем, при котором сбои полностью исключены

Г. Это метод разработки программных систем, при котором информация о сбоях защищена от пользователя

Пункт 2. Какие методы разработки программного кода применяются в защитном программировании?

\*А. Внедрение в программный код проверок на допустимость данных

\*Б. Внедрение в программный код функций сбора информации о сбоях

В. Ограничение функциональности системы для исключения аварий

\*Г. Внедрение в программный код функций восстановления после сбоев

Пункт 3. Какие механизмы защитного программирования применяются в языках высокого уровня?

А. Обработка прерываний

\*Б. Обработка исключений

\*В. Проверка допущений

Г. Проверка отключений

**Вопрос 6. Допущения**

Пункт 1. В каких случаях рекомендуется использовать допущения в критических точках?

\*А. В качестве помощи при отладке

\*Б. В качестве помощи при документировании

\*В. В качестве помощи при формальной инспекции алгоритмов

Г. В качестве помощи при компиляции кода

Пункт 2. Какие существуют типы допущений?

\*А. Предусловия

\*Б. Постусловия

В. Инусловия

\*Г. Инварианты

Пункт 3. Какие типы допущений являются основой для контрактного программирования?

\*А. Предусловия

Б. Постусловия

В. Инусловия

Г. Инварианты

**Вопрос 7. Обработка исключительных ситуаций**

Пункт 1. Что такое исключение?

\*А. Аномальная ситуация, приводящая к тому, что нормальное выполнение программной системы становится невозможным

Б. Аномальная ситуация, вызванная действиями пользователя

В. Аномальная ситуация, не предусмотренная требованиями

Г. Аномальная ситуация, вызывающая сообщения об ошибке

Пункт 2. Какие причины могут приводить к возникновению исключений?

\*А. Предусловие не выполняется на входе в блок

\*Б. Постусловие не выполняется на выходе из блока

\*В. Инвариант класса не выполняется на входе или выходе

\*Г. Инвариант цикла не выполняется на итерации цикла

Пункт 3. Какие методы применяются для тестирования исключений?

\*А. Явная генерация исключений

\*Б. Приведение системы в состояние, ведущее к исключению

\*В. Анализ кода

Г. Уменьшение количества исключений

**Вопрос 8. Сбор информации о сбоях и отказах**

Пункт 1. Что может служить источником информации о сбое работающей системы?

\*А. Сообщение пользователя с описанием сбоя

Б. Сообщение от аварийно завершившейся системы

\*В. Сообщение от подсистемы оповещения о сбоях

Г. Сообщение от тестировщика

Пункт 2. В чем отличие отчета о сбое, созданного пользователем от отчета о сбое, созданного разработчиком?

\*А. В нем меньше технических деталей

Б. В нем точно указано местоположение дефекта

В. В нем указаны причины сбоя

Г. В нем отсутствует информация о времени сбоя

Пункт 3. Какие подходы могут применяться для получения необходимых технических деталей о сбое?

\*А. Автоматизированный сбор информации о сбое

\*Б. Опрос пользователя техническим специалистом

В. Предоставление пользователю проектной документации

\*Г. Выполнение пользователем процедуры сбора информации, описанной в документации

**Раздел. «Промышленная разработка программного обеспечения»**

**Вопрос 1. Конфигурационное управление**

Пункт 1. Укажите основные задачи процесса конфигурационного управления

\*А. Обеспечение целостности проектных данных

\*Б. Поддержка взаимодействия между членами коллектива разработчиков

В. Обеспечение качества разрабатываемой системы

Г. Управление проектом

Пункт 2. Укажите основные составные части процесса конфигурационного управления

\*А. Управление версиями

\*Б. Управление состояними

\*В. Управление выпуском

Г. Управление работами

Пункт 3. Процедуры процесса управления конфигурациями включают в себя

\*А. Управление изменениями

\*Б. Формирование базовых конфигураций

\*В. Аудиты конфигураций

\*Г. Резервное копирование конфигураций

**Вопрос 2. Идентификация объектов конфигурации**

Пункт 1. Какую цель имеет процедура идентификации объектов конфигурации?

\*А. Присвоение каждому объекту конфигурации уникального имени, позволяющего отличать его от других

Б. Определение процедуры идентификации пользователей для доступа к объектам

В. Определение схемы аудита идентифицированных объектов конфигурации

Г. Предотвращение неверных правил доступа к объектам конфигурации

Пункт 2. Каким образом именуются объекты конфигурации?

\*А. При помощи процедуры присвоения имен

\*Б. При помощи задания нового уникального идентификатора

В. Путем составления индексов конфигураций

Г. Путем образования иерархии объектов

Пункт 3. Для каких процедур конфигурационного управления необходима идентификация объектов конфигурации?

\*А. Для трассировки

\*Б. Для управления состояниями

\*В. Для аудита

\*Г. Для архивирования

**Вопрос 3. Управление изменениями**

Пункт 1. Какова цель процедуры управления изменениями?

А. Обеспечение возможности хранить изменения независимо от данных проекта

\*Б. Обеспечения контроля управляемости изменений, вносимых в систему и целостности системы после внесения изменений

В. Создание отчетов для аудита процесса управления конфигурациями

Г. Поддержка процесса архивации

Пункт 2. Каким образом вносятся изменения в конфигурацию?

\*А. По результатам утверждения запроса на изменение менеджером конфигурации

Б. По желанию разработчиков

В. По запросу от пользователей

Г. По запросу от руководителя проекта

Пункт 3. В какие конфигурации запрещено внесение изменений?

А. В рабочие конфигурации

\*Б. В базовые конфигурации

\*В. В конфигурации, предназначенные для сертификации

Г. В личные конфигурации

**Вопрос 4. Аудиты конфигураций**

Пункт 1. Какие типы аудита конфигураций вы знаете?

А. Аудит компетентности менеджера конфигурации

\*Б. Аудит процессов управления конфигурациями

\*В. Аудит объектов конфигураций

Г. Аудит тестирования конфигураций

Пункт 2. Какие вопросы проверяются в ходе аудита конфигураций?

\*А. Проведены ли изменения в соответствии со стандартами проекта?

Б. Проведены ли изменения в соответствии с указаниями руководителя проекта?

\*В. Корректно ли поименованы все объекты конфигурации?

\*Г. Все ли необходимые объекты присутствуют в базовых конфигурациях?

Пункт 3. Какие аспекты конфигурационного управления могут подвергаться аудиту?

\*А. Идентификация объектов

\*Б. Управление состояниями

\*В. Управление выпуском

\*Г. Управление версиями

**Вопрос 5. Стандарты качества программного обеспечения**

Пункт 1. Какова основная цель стандартов качества?

А. Определить характеристики продукции, удовлетворяющие критериям качества

\*Б. Определить процессы выпуска продукции, гарантирующие выпуск продукции, удовлетворяющей заданным критериям качества

В. Определить конкретный набор типов выпускаемой продукции, которая может быть выпущена качественно исходя из текущего состояния процессов предприятия

Г. Определить набор процессов предприятия, который может быть использован при выпуске качественной продукции

Пункт 2. Какие типы процессов разработки обычно покрывают стандарты качества?

\*А. Основные процессы разработки

\*Б. Обеспечивающие процессы

\*В. Процесс планирования

\*Г. Процесс анализа рисков

Пункт 3. Каким образом гарантируется качество проектов по разработке ПО?

\*А. Определяются технологические процессы, гарантирующие повторяемость процесса разработки и соответствующие требованиям стандартов качества

Б. Определяются ответственные за качество продукции

В. Определяются аудиторы качества

Г. Определяются средства измерения ответственности персонала

**Вопрос 6. Стандарты качества серии ISO 9000**

Пункт 1. Какие принципы лежат в основе ISO 9000?

\*А. Ориентация на потребителя

Б. Ориентация на проектную команду

\*В. Лидерство руководителя

Г. Ответственность персонала

Пункт 2. Какие принципы лежат в основе ISO 9000?

\*А. Вовлечение работников

\*Б. Процессный подход

В. Измерения процессов

Г. Постоянное упрощение схем взаимодействия

Пункт 3. Какие принципы лежат в основе ISO 9000?

\*А. Принятие решений, основанное на фактах

Б. Своевременная передача данных потребителю

\*В. Постоянное улучшение

\*Г. Системный подход к управлению

**Вопрос 7. Стандарты качества серии ISO 9000**

Пункт 1. Какие процессы верхнего уровня определяет ISO 9000?

\*А. Система менеджмента качества

\*Б. Ответственность руководства

В. Планирование задач

Г. Управление жизненным циклом

Пункт 2. Какие процессы верхнего уровня определяет ISO 9000?

\*А. Управление ресурсами

Б. Анализ коммерческих предложений

\*В. Выпуск продукции

Г. Вывод системы из эксплуатации

Пункт 3. Какие процессы верхнего уровня определяет ISO 9000?

\*А. Измерения

Б. Управление контрактами

В. Управление клиентами

\*Г. Анализ и улучшения

**Вопрос 8. Стандарты качества серии ISO 9000**

Пункт 1. Какие документы создаются в результате аудита системы качества?

\*А. Отчеты о несоответствии

\*Б. Отчеты о наблюдении

\*В. Заключение аудитора

Г. Отчеты о изменениях

Пункт 2. Какие виды действий предпринимаются компанией по результатам аудита системы качества?

\*А. Корректирующее действие

\*Б. Коррекция

В. Предупреждающее действие

Г. Обновляющее действие

Пункт 3. Какие виды аудитов системы менеджмента качества вы знаете?

\*А. Первичный

\*Б. Внутренний

\*В. Надзорный

Г. Регулятивный