Вопросы к заочному туру олимпиады «Инфотелеком» (НИИ «Масштаб»)

1. Невероятно, но факт: 1/8 > 1/4. Вот и доказательство:

|  |  |
| --- | --- |
| 1 | 3 > 2 |
| 2 | 3 \* log10(1/2) > 2 \* log10(1/2) |
| 3 | log10(1/2)3 > log10(1/2)2 |
| 4 | (1/2)3 > (1/2)2 |
| 5 | 1/8 > 1/4 |

Или же в данное доказательство закралась ошибка? Среди предложенных выберите верное утверждение, корректно описывающее сделанную в доказательстве ошибку.

**Варианты ответов**:

1. Во второй строке обе части неравенства были умножены на отрицательную величину, но знак неравенства при этом не поменялся.
2. В третьей строке при возведении в степень подлогарифмического выражения нужно было изменить знак неравенства, т.к. возводимое в степень число меньше единицы.
3. В четвёртой строке знак неравенства должен был измениться, т.к. при сравнении не было учтено основание логарифма.
4. Ошибка произошла как в строке 2, так и в строке 3 (см. варианты А и В).

**Пояснения.** Т.к. log x < 0 для 0 < x < 1, то поскольку обе части неравенства 3 > 2 умножены на негативную величину log10(1/2), знак > во второй строке доказательства должен быть заменен на <.

1. Невероятно, но факт: если a и b – два равных вещественных числа, то a = 0. Вот и доказательство:

|  |  |
| --- | --- |
| 1 | a = b |
| 2 | a2 = a \* b |
| 3 | a2 – b2 = a \* b – b2 |
| 4 | (a – b) \* (a + b) = (a – b) \* b |
| 5 | a + b = b |
| 6 | a = 0 |

Или же в данное доказательство закралась ошибка? Среди предложенных выберите верное утверждение, корректно описывающее сделанную в доказательстве ошибку.

**Варианты ответов**:

1. В строке 2 при умножении на величину «а» обеих частей уравнения некорректно изменилась область допустимых значений параметра a.
2. В строке 3 при вычитании квадрата b из обеих частей уравнения не было учтено значение дискриминанта соответствующего квадратного уравнения.
3. В строке 4 нельзя было делить обе части уравнения на число (a – b), т.к. эта величина всегда равна нулю по условию.
4. В строке 5 нельзя было сокращать обе части уравнения на величину b, т.к. область допустимых значений этой величины не принадлежит полю комплексных чисел.

**Пояснения.** Ошибка в четвертой-пятой строках доказательства: мы не можем сократить разность (a – b), т.к. a – b = 0.

1. Вычисление угла между стрелками часов (минутной и часовой). Стрелочные часы на стене показывают время 18:30. Каков угол между минутной и часовой стрелками (меньший угол)?

**Варианты ответов**:

1. π/6 радиан.
2. 15 о.
3. 5 о.
4. 5 градов.

**Пояснения.** Ключ к решению задачи – перевод позиции каждой стрелки в угол и вычесть один угол из другого для получения разницы. Далее – по шагам:

* Для минутной стрелки: каждая минута – 6о (всего в циферблате 360 о и один час (=60 минут): 360/60 = 6 о), позиция минутной стрелки: 6 о \* минуты (например, минутная стрелка на 12, тогда угол равен: 6 о \* 0 = 0 о)
* Для часовой стрелки: каждая минута – 0.5 о (всего в циферблате 360 о и 12 часов (= 720 минут), одна минута для часовой стрелки: 360/720 = 0.5 о), позиция часовой стрелки: 0.5 о \* минуты (например, часовая стрелка на 2 часах: 0.5 о \* 120 = 60 о)
* 18:30 – или 6:30 – переводим в минуты: 6 \* 60 + 30 = 390 минут, для часовой стрелки угол будет: 390 \* 0.5 о = 195 о , для минутной стрелки, очевидно, угол будет 180 о , т.о. получаем ответ: 195 о – 180 о = 15 о.

1. Кажется логичным? Необходимо указать число, которое должно быть на месте знака «?».

8080 = 6

1357 = 0

2022 = 1

1999 = 3

6666 = ?

**Варианты ответов**:

1. Один.
2. Два.
3. Четыре.
4. Восемь.

**Пояснения.** Число 4 – количество окружностей в начертании цифр, составляющих число.

1. Как определить, применяя только побитовые и арифметические операции, является ли целое положительное число степенью числа 2.

**Варианты ответов**:

1. Нужно применить побитовую конъюнкцию проверяемого числа и числа, на единицу его меньшего. Затем проанализировать результат.
2. Нужно в цикле сложить все единицы в двоичном представлении числа и проанализировать результат, т.е. без применения цикла поставленную задачу решить невозможно.
3. Нужно взять двоичный логарифм проверяемого числа и разделить на натуральный логарифм этого же числа. Затем проанализировать, является ли результат целым числом.
4. Нужно рассчитать гиперболический синус проверяемого числа, затем вычислить значение натурального логарифма и проанализировать, является ли полученный результат целым положительным числом, если к нему применена операция дизъюнкции с исходным числом.

**Пояснения.** Ключ к ответу лежит в понимании двух фактов:

* Любая степень 2 в двоичной системе счисления начинается с 1, все остальные разряды – 0.
* При вычитании 1 из любого числа, являющегося степенью 2, получившееся число в двоичном представлении содержит все единицы

Пример1: является ли 16 степенью 2? Пояснение: 16 (десятичная) = 10000 (двоичная). 16 – 1 = 15 (десятичная) = 01111 (двоичная). Применяя побитовую операцию получаем: 10000 & 01111 = 00000, это значит 16 является степенью 2.

Пример2: является ли 15 степенью 2? Пояснение: 15 (десятичная) = 1111(двоичная). 15 – 1 = 14 (десятичная) = 1110 (двоичная). 1111 & 1110 = 1110. 15 не является степенью 2.

1. В параллельной вселенной программисты додумались, как и мы, хранить и передавать числа с помощью бинарных кодов, состоящих из нолей и единиц. Однако при этом наши антиподы используют гипотетическую фибоначчиевую систему счисления вместо привычной нам двоичной. В результате, у них код «1100» означает не двенадцать, а восемь. Каким образом можно передать в параллельную вселенную сообщение, содержащее ответ на главный вопрос жизни (т.е. передать число 42)?

**Варианты ответов**:

1. 11010100.
2. 11000100
3. 11101110.
4. 10010000.

**Пояснения.** Правильным ответом являются коды 10010000, 10001100, 10001011, 1110000, 1101100, 1101011, каждый из которых является корректной записью числа 42. Участники олимпиады должны по подсказке в задании (1100b=8Ф) догадаться, что в фибоначчиевой системе счисления вместо двоичных весов разрядов «1, 2, 4, 8, 16…» используются числа Фиббоначи «1, 2, 3, 5, 8, …». Если участник олимпиады привёл только один из указанных кодов, то поставить ему 50% баллов. Если привёл от двух до четырёх, поставить 90%. Если привёл от пяти до шести, поставить 100%.

1. Гламурная блондинка Ефросинья купила смартфон и установила в него две SIM-карты разных операторов связи в надежде, что теперь сможет смотреть любимые онлайн-сериалы с суммарной скоростью двух Интернет-подключений (по 1 Мбит/с каждое). Однако при попытке вручную выполнить пакетную балансировку IP-пакетов между двумя подключениями она выяснила, что TCP-приложения получили не сумму пропускных способностей (2 Мбит/с), а всего лишь 500 кбит/с, т.е. меньше, чем если бы она использовала всего один Интернет-канал без всякой балансировки. Чем это можно объяснить?

**Варианты ответов**:

1. ТСР-пакеты при балансировке направлялись в оба канала связи в виде дубликатов для обеспечения высокой надёжности и помехоустойчивости передачи.
2. Ефросинье следовало использовать протокол UDP вместо TCP, т.к. UDP управляет размером окна при наличии нескольких соединений.
3. ТСР и все существующие протоколы не рассчитаны на работы в multihome-среде, т.к. возникают сложности с обеспечение корректной маршрутизации.
4. Используемые Ефросиньей Интернет-подключения имели существенно асимметричные значения RTT, что не позволило протоколу ТСР корректно выбрать размер окна при пакетной балансировке.

**Пояснения.** Проблема могла возникнуть, только если каналы связи Ефросиньи имели асимметричные характеристики (например, в одном канале RTT-задержка равна 500 мс, а в другом 10 мс или в одном потери 1%, а в другом 10% и т.п.). Причина проблемы – в особенностях реализации механизма окна в ТСР: окно позволяет подстроить скорость передачи только под стабильный канал связи, в котором характеристики почти не меняются (т.е. RTT почти постоянен и мало колеблется). А при пакетной балансировке половина пакетов идут по каналу с большой RTT, а другая половина – по каналу с маленькой RTT, что не позволяет TCP правильно настроить окно. Первый способ частично решить проблему Ефросиньи – это использовать балансировку по потокам, а не по пакетам (тогда каждый ТСР-поток будет все свои пакеты отправлять по одинаковому пути). Второй способ решить проблему – использовать вместо ТСР протоколы SCTP, MP-TCP, в которых каждый используемый канал связи настраивается независимо от другого, что позволяет успешно агрегировать пропускную способность нескольких каналов связи. Если олимпиадник приведёт хотя бы один из этих вариантов решения проблемы, стоит зачесть ответ.

1. Известный оператор связи изменил способ тарификации Интернет-трафика необычным образом: во всех рекламных брошюрах были изменены только единицы измерения – вместо килобайтов стали фигурировать кибибайты (KiB), вместо мегабайтов – мебибайты (MiB) и так далее. Например, если раньше за 100 рублей можно было скачать из Интернета 100 гигабайт лицензионного программного обеспечения, то теперь за те же 100 рублей можно скачать 100 гибибайт. Стал ли трафик стоить пользователям дороже или дешевле (поясните ответ, 50% баллов)?

**Варианты ответов**:

1. Трафик подорожал, т.к. каждый кибибайт (мебибайт, гибибайт) при передаче занимает меньшую полосу пропускания.
2. Трафик подешевел, т.к. за те же деньги пользователи стали получать больше данных.
3. Стоимость трафика не изменилась, т.к. изменение единиц измерения – обычный маркетинговый ход для несведущих пользователей.
4. Те тарифы, где килобайты были заменены на кибибайты, подешевели. Остальные – подорожали.

**Пояснения.** Трафик подешевел. Один кибибайт равен 1024 байт, что больше, чем 1 килобайт (килобайт равен 1000 байт, см. <https://ru.wikipedia.org/wiki/IEEE_1541-2002>). Аналогично, 1 мебибайт равен 2^20=1048576 байт > 1 мегабайт (и так далее). Значит, за те же деньги провайдер разрешил пользователям скачать больше данных, т.е. трафик подешевел.

1. Два whitehat-хакера – Прокофий и Онуфрий – никогда не выходят в открытый Интернет и общаются только с использованием tor-браузера в Твиттере. Прокофий лжёт только по понедельникам, вторникам и средам, а в остальные дни говорит только правду. Онуфрий лжёт по четвергам, пятницам и субботам, говоря только правду в остальные дни. Сегодня у Прокофия появился твит: «Вчера я солгал». Онуфрий ответил ему: «Я тоже». Помогите специалистам управления «К» МВД РФ определить день недели, когда произошла эта переписка.

**Варианты ответов**:

1. Понедельник.
2. Вторник.
3. Среда.
4. Четверг.

**Пояснения.** Четверг (можно подобрать методом исключения и перебора).

1. Программист очнулся в вагоне поезда, состоящего из большого количества одинаковых вагонов, между которыми можно свободно переходить. Кондуктор сообщил программисту, что поезд находится на круговой железной дороге и первый вагон соединяется с последним. В каждом вагоне есть стоп-кран, который может находиться в одном из двух положений, но начальное состояние стоп-кранов неизвестно. После десятой кружки кофе в вагоне-ресторане программист придумал алгоритм, позволяющий посчитать количество вагонов в поезде. Опишите этот алгоритм (словесно или на любом языке программирования), зная, что программисту разрешено использовать стоп-краны?

**Варианты ответов**:

1. Нужно идти всегда в одну сторону, опуская все стоп-краны. При обнаружении зацикливания нужно посчитать количество пройденных вагонов с учётом вычета тех из них, которые программист изначально застал в опущенном состоянии.
2. Нужно идти всегда в одну сторону, опуская стоп-краны в чётных поездах и поднимая в нечётных. При обнаружении зацикливании посчитать количество пройденных вагонов и умножить на два (для чётного случая) или умножить на два и прибавить один (для нечётного случая).
3. Нужно попеременно идти то влево, то вправо от исходного вагона с каждым разом увеличивая количество проходимых вагонов, при этом поднимая стоп-краны в одном направлении и опуская в противоположном. При обнаружении неверного положения стоп-крана в посещаемом вагоне нужно остановиться и посчитать количество вагонов.
4. Нужно пройти несколько оборотов по поезду, запоминая расположения всех стоп-кранов. Затем усреднить количество посчитанных вагонов, рассчитав среднее количество обнаруженных в поднятом положении стоп-кранов, умноженное на два.

**Пояснения.** Идя налево от исходного вагона, программист опускает стоп-краны в посещаемых вагонах; идя направо – поднимает. При этом он поочерёдно двигается то налево, то направо, увеличивая на один количество проходимых вагонов, считая от исходного в каждую сторону. Для этого он держит в уме пару чисел i и j, которые изменяются следующим образом: +1, -1, +2, -2, +3, -3, …, где знак означает направление движения (налево или направо), а числа означают количество проходимых вагонов, считая от исходного. Программист точно знает количество пройденных вагонов и знает, какое он в них установил положение стоп-кранов. По мере увеличения i и j в какой-то момент самый левый посещённый вагон совпадёт с самым правым (круг замкнётся), при этом программист обнаружит неверное положение стоп-крана в уже пройденном им вагоне. Для подсчёта количества вагонов достаточно будет в этот момент сложить модули переменных i и j.

1. Встретились два программиста на дискотеке:

– Сколько у тебя компьютеров?

– Три. К сожалению, все однопроцессорные.

– А сколько ядер на процессорах?

– Если количества ядер перемножить, получится 36.

– Мало информации, не могу посчитать.

– Суммарное количество ядер равно возрасту ди-джея.

– Всё еще мало информации, хотя ди-джей мой родной брат.

– Самый слабый процессор не поддерживает SSE4.

– О, теперь знаю!

После этого был правильно назван ответ (количество ядер в процессорах). Каков этот ответ?

**Варианты ответов**:

1. 1, 6, 6.
2. 2, 3, 6.
3. 1, 4, 9.
4. 1, 2, 18.

**Пояснения.** 1, 6, 6 (или любая перестановка этих чисел). Из условия понятно, что решение надо искать в целых числах. Зная делители числа 36 (1, 2, 3, 4, 6, 9, 12, 18), можно методом перебора найти уникальные разложения на множители числа 36, см. в таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Проц1 | Проц2 | Проц3 | Сумма | Произведение |
| 1 | 1 | 36 | 38 | 36 |
| 1 | 2 | 18 | 21 | 36 |
| 1 | 3 | 12 | 16 | 36 |
| 1 | 4 | 9 | 14 | 36 |
| 1 | 6 | 6 | 13 | 36 |
| 2 | 2 | 9 | 13 | 36 |
| 2 | 3 | 6 | 11 | 36 |
| 3 | 3 | 4 | 10 | 36 |

Уточнение про ди-джея будет недостаточным для ответа, только если сумма ядер равна повторяющемуся в таблице числу 13 (иначе сразу после этого уточнения был бы найден ответ по таблице, т.к. возраст брата точно известен). Третье уточнение позволяет отбросить вариант 2, 2, 9, т.к. в нём присутствуют сразу два одинаково слабых процессора. Остаётся ответ. 1, 6, 6.

1. Вольнонаёмный работник Ахмед соединил два компьютера медным проводом и собирается передавать по этому каналу связи двоичные данные, кодируя «единицу» и «ноль» напряжением электрического тока в 10 В и 5 В соответственно, реализуя тем самым известный метод физического кодирования NRZ. Какие сложности ожидают Ахмеда при декодировании полученного NRZ-сигнала при условии, что для синхронизации таймеров на приёмо-передающих устройствах не посылаются синхроимпульсы и что при передаче возникают заметные помехи от включаемой Ахмедом в обеденное время микроволной печи, хотя при этом Ахмед не забыл сделать гальваническую развязку при подключении провода к компьютеру.

**Варианты ответов**:

1. Будут часто возникать ошибки при передаче длинных последовательностей нулей или единиц, т.к. сигнал в этом случае выродится в постоянный.
2. В NRZ сложно различить высокий и низкий уровень сигналов при наличии гальванической развязки.
3. Таймеры рассинхронизируются при наличии помех, что приведёт к высоким потерям распознанных сигналов даже при использовании гальванической развязки.
4. Занимаемая NRZ полоса пропускания канала прямо пропорциональна уровню помех, что не позволит Ахмеду корректно использовать гальваническую развязку при рассинхронизированных приёмо-передатчиках.

**Пояснения.** Ответ можно зачесть, если студент даст хотя бы один приближённо сформулированный ответ из следующих трёх: 1) в NRZ возникают ошибки распознавания длинных последовательностей нулей или единиц из-за накапливающейся рассинхронизации; 2) в NRZ отсутствует автоматическое обнаружения ошибок передачи (в других методах есть запрещённые уровни или комбинации сигналов, получение которых позволяет автоматически распознать возникшую в результате помехи ошибку); 3) при передаче длинных последовательностей нулей или единиц NRZ вырождается в сигнала постоянного тока, а значит он не сможет преодолеть гальваническую развязку, обычно используемую при подключении компьютера к сети. Уточнение по первому пункту: отсутствие синхронизации приёмо-передатчиков приводит к тому, что при передаче длинных последовательностей нулей («00000000…») или единиц («1111…111») приёмник может принять неверное их количество, т.к. расхождение часов приёмника с часами передатчика с каждым полученным битом в этом случае накапливается. При чередовании 0 и 1 приёмник синхронизирует свои часы по фронту изменяющегося сигнала. Это становится невозможно сделать, если сигнал долго не меняется.

1. Число 201620162016 записано в системе счисления с основанием 343. Покажите, как будет выглядеть его запись в семеричной системе счисления (ответ поясните).

**Варианты ответов**:

1. 201620162016\*343/7.
2. 20001060200010602000106\*(343\*7).
3. 2000001006002000001006002000001006\*343/7/7/7.
4. .

**Пояснения.** 2 000 001 006 002 000 001 006 002 000 001 006 (пробелы между цифрами не обязательны и нужны просто для удобства чтения). Участник должен дать пояснения, что 343 – это 7 в кубе, и поэтому можно переводить числа из одной системы в другую просто заменяя одну цифру в 343-чной системе счисления на три соответствующие цифры в семеричной системе счисления (по аналогии с переводом восьмеричных чисел в двоичные).

1. Выберите из приведённых фраз корректные утверждения, относящиеся к определению понятия функция-замыкание (function closure).

**Варианты ответов**:

1. Функция, являющая методом класса, при условии, что в ней используются статические члены класса для хранения разделяемого состояния.
2. Функция, содержащая ссылки на глобальные переменные, которые «замыкаются» в теле рассматриваемой функции для дальнейшего использования.
3. Функция с рекурсивным вызовом при концевом замыкании рекурсии.
4. Функция, в которой используются переменные, объявленные вне тела этой функции и не являющиеся её параметрами.

**Пояснения.** Вот цитата с Википедии с объяснением функции-замыкания – «функция первого класса, в теле которой присутствуют ссылки на переменные, объявленные вне тела этой функции в окружающем коде и не являющиеся её параметрами. Замыкание есть способ представления функциональности и данных, связанных и упакованных вместе. Функция-замыкание определена в теле другой функции и создаётся каждый раз во время её выполнения. Это выглядит как функция, находящаяся целиком в теле другой функции. При этом вложенная внутренняя функция содержит ссылки на локальные переменные внешней функции». Пример на Python:

def outer(init):

a = init   
 def inner(b):

return b + a # функция inner замкнула в себе переменную a (т.е. запомнила её)

return inner

f = outer (1)

f(10) # на момент этого вызова а как локальная переменная в outer уже не существует,

11 # но при этом inner вполне корректно «видит» её значение и считает сумму (a+b)

g = outer(3) # новая копия переменной a замыкается теперь в g

g(5)

8 # замыкания f и g обладают своей собственной независимой копией a

1. В 8-разрядную ячейку памяти можно записать любое целое число от -128 до +127, если использовать для представления отрицательных чисел дополнительный код. Как бы изменился этот диапазон, если для представления знаковых целых чисел использовалась бы система счисления с основанием «‑2» (минус два)? Ответ поясните.

**Варианты ответов**:

1. От -170 до +85.
2. От -128 до +127.
3. От -127 до +127.
4. От 0 до +255.

**Пояснения.** Новый диапазон содержит числа от -170 до 85. Минимальное число – это 10101010, максимальное число – это 01010101. Знаки разрядов чередуются, т.к. -2 возводится то в чётную степень, то в нечётную.

1. В заголовке пакета IP (версия 4) для указания размера пакета выделено 16 бит, что позволяет передавать пакеты размером вплоть до 65535 байт. Однако, наблюдая за размерами реально курсирующих IP-пакетов в Интернете, можно убедиться, что их размеры практически никогда не превышают 2000 байт. Чем это можно объяснить?

**Варианты ответов**:

1. Наиболее часто IP-пакеты передаются по Ethernet-сетям, в которых предельно допустимый размер фрагмента передаваемых данных равен примерно 1500 байт.
2. При передаче пакетов большего размера наблюдается падение производительности таких протоколов, как TCP (SCTP) вследствие увеличения количества повторных передач при переполнении буфера 2000 байт.
3. Размер MTU для протокола IP составляет 1518 байт, хотя в заголовке пакета выделено целых 16 бит, часть из которых зарезервированы для развития протокола в будующем.
4. Это является особенностью архитектуры протокола IP версии 4. В IP версии 6 указанная проблема не наблюдается, т.к. в нём предельный размер MTU увеличен с 2000 байт до 16000 байт.

**Пояснения.** Наиболее часто IPv4-пакеты в современных сетях передаются по Ethernet-сетям, в которых предельно допустимый размер фрагмента данных (кадр) равен примерно 1500 байт (значение MTU). Поэтому при попытке передать IP-пакет более 1500 байт он будет фрагментирован на несколько кадров по 1500 байт, каждый из которых будет передан независимо. Затем на другом конце эти фрагменты придётся собирать воедино. Все эти операции по фрагментированию и сборке являются нежелательными (замедляется RTT), поэтому стараются их избегать, не давая размеру пакетов выходить за 1500.

1. В массиве из 11 элементов записаны целые 32-разрядные числа. Среди них есть пять пар неизвестных дубликатов и одно неизвестное произвольное число без пары. Приведите схему алгоритма (можно в виде текста программы на любом языке программирования) для поиска числа без пары за один проход массива в цикле. Например, если числа в массиве таковы: «1, 2, 1, 3, 4, 2016, 5, 2, 5, 4, 3», то результатом работы алгоритма должно стать число 2016.

**Варианты ответов**:

1. К элементам массива применяется операция хеширования произвольной функцией с последующим побитовым and (конъюнкция).
2. К элементам массива применяется операция filter в виде побитового or (дизъюнкция).
3. К элементам массива применяется операция map в виде побитового and (конъюнкция).
4. К элементам массива применяется операция reduce в виде побитового xor (исключающее или).

**Пояснения.** В цикле ко всем элементам массива попарно применяется побитовая операция xor (исключающее или) с накоплением результата в переменной. В результате числа-дубли друг друга уничтожат (т.к. (x xor x) = 0 для любого х), а уникальное число без пары сохранит свои биты и останется в накопленной переменной.

1. Требуется на заданном языке программирования написать две функции для вычисления минимума и максимума из двух заданных целых чисел max(a, b) и min(a, b) без использования знаков сравнения (<, > и т.п.), а также операторов if, case и тернарной условной операции.

**Варианты ответов**:

1. Написать указанную функцию невозможно на языках, более высокоуровневых, чем язык Ассемблера.
2. Поставленная задача решаема только в функциональных языках программирования.
3. Задача может быть решена только в тех языках программирования, где при заданных ограничениях реализуема функция вычисления модуля числа.
4. Возможное решение задачи будет иметь ограничение на ширину разрядной сетки, используемой для представления целых чисел на целевой платформе.

**Пояснения.** max(a,b) = (a/2 + b/2 + abs(a/2 - b/2)), min(a,b) = (a/2 + b/2 - abs(a/2 - b/2))

1. Выберите из предложенных вариантов несуществующий или неверно описанный шаблон проектирования программного обеспечения, применяемый при объектно-ориентированном программировании.

**Варианты ответов**:

1. Декоратор (Decorator) – класс, расширяющий функциональность другого класса без использования наследования.
2. Заместитель (Proxy) – класс, который является посредником между двумя другими объектами.
3. Хранитель (Memento) – класс, позволяющий зафиксировать и сохранить внутреннее состояние объекта, чтобы позднее восстановить его.
4. Строитель (Builder) – класс, который позволяет построить ровно один экземпляр класса, инициализирующийся в момент первого обращения к нему.
5. В односвязном списке из n элементов записаны целые числа от 1 до n в произвольном порядке (каждое число встречается ровно один раз). Один элемент списка случайным образом удаляется. Каким образом можно за один проход полученного списка обнаружить удалённый элемент?

**Варианты ответов**:

1. Нужно отсортировать элементы списка по убыванию, а затем найти элемент, который идёт не по порядку (т.е. с пропуском). Искомый элемент будет на единицу меньше.
2. Нужно выполнить операцию побитового исключающего или со всеми элементами списка и вычесть из неё побитовую инверсию суммы этих элементов.
3. Нужно отсортировать элементы списка по возрастанию и искать наименьший непрерывно расположенный элемент, который при вычитании от размера списка даст номер искомого удалённого элемента.
4. Нужно из суммы арифметической прогрессии от 1 до n (с шагом 1) вычесть сумму элементов списка, рассчитанную при однократном проходе списка.
5. Выберите список, в котором присутствует несуществующее название языка программирования.

**Варианты ответов**:

1. Ada, Algol, Cobol, Go, D.
2. Erlang, F#, Tcl, Haskell, Rust.
3. Lisp, Smalltalk, Eslaf, Scala, Lua.
4. Swift, Groovy, Prolog, Brainfuck, R.
5. Всемирная организация по стандартизации (ISO) уже около сорока лет назад предложила сетевую модель OSI для организации взаимодействия в компьютерных сетях. Однако на данный момент в мире не существует ни один стек протоколов и ни один протокол, который бы точно реализовывал модель OSI. Выберите **ложное** утверждение среди приведённых.

**Варианты ответов**:

1. В протоколе TCP реализуются как функции сетевого, так и транспортного уровня.
2. В протоколе UDP реализуются те только функции транспортного уровня.
3. В протоколе FTP реализованы функции как минимум трёх уровней OSI-модели.
4. В протоколе SSH аутентификация реализована как на канальном уровне, так и на прикладном.
5. Иногда вместо использования библиотеки динамической компоновки (т.е. файлы \*.DLL и \*.SO) следует предпочесть использовать эту библиотеку в виде статической компоновки (т.е. файлов «прилинковать» файлы \*.LIB и \*.a). Укажите причину, которая \_**не\_** является достаточным основанием для этого.

**Варианты ответов**:

1. В программе существуют глобальные статические переменные, которые при динамической линковке становятся разделяемыми.
2. Существует ровно одно приложение, которое будет использовать библиотеку.
3. Программист знает, что на целевой системе наблюдается проблема “DLL Hell” в отношении к рассматриваемой библиотеке.
4. Компилятор или операционная система не поддерживает динамическую линковку для библиотек.

**Пояснения.** Ниже приведён неполный перечень правильный ответов. Если участник олимпиады укажет любую из них, можно засчитывать вопрос за правильный.

Ситуация 1. Если библиотека будет использована только одним приложением, поэтому динамическая линковка не сможет проявить свою самую сильную сторону, т.е. быть разделяемой несколькими приложениями для экономии памяти. Кроме того, в этом случае при статической линковке из библиотеки будут добавлены в исполняемый файл только нужные и используемые модули, в отличие от DLL, которая в любом случае загружается в память полностью.

Ситуация 2. Планируется использовать встроенную или урезанную операционную систему, которая не поддерживает возможность динамической линковки как таковую.

Ситуация 3. Используемая библиотека является нестабильной и приходится для её корректной работы использовать недокументированные возможности и вручную обходить проблемные места. В этом случае использование динамического связывания добавляет риск того, что пользователь или ОС обновят DLL без ведома программистов, в результате его работа может нарушится из-за изменения недокументированных особенностей.

Ситуация 4. В системе наблюдается проблема «DLL Hell» и её необходимо решить.

Ситуация 5. Программа используется в системе реального времени с очень жёсткими требованиями ко времени первого запуска приложения. При использовании DLL время первого запуска может быть несколько больше, чем при использовании статически слинкованной библиотеки.

1. В какой ситуации операция ++i может работать в объектно-ориентированном языке программирования быстрее, чем операция i++?

**Варианты ответов**:

1. При выполнении i++ создаётся (в отличие от ++i) дополнительная копия объёмного объекта i на всё время выполнения выражения, в котором участвует i++.
2. При выполнении ++i компилятор создаёт дополнительную копию объекта i на время выполнения выражения, что позволяет оптимизировать операцию инкремента.
3. При выполнении ++i может наблюдаться незначительный прирост производительности при оптимизации на уровне Ассемблера, иначе во всех случаях обе операции работают сравнимо быстро.
4. При выполнении i++ компилятор анализирует тело цикла, в котором участвует итератор и оптимизирует его применение с учётом иерархии наследования класса объекта i.

**Пояснения.** При выполнении i++ создаётся дополнительная копия начального значения i на всё время выполнения выражения, в котором участвует i++, т.к. инкремент i должен произойти после вычисления всего выражения. При выполнении ++i этого не происходит, т.к. в выражении участвует уже изменённое новое значение i, а значит нет смысла создавать промежуточную копию. В результате, если i является громоздким объектом, то при создании его доп. копии в операции i++ может уйти намного больше времени, чем в операции ++i. Если же i – простая переменная, то эти накладные расходы будут ничтожно малы и ими можно было бы пренебречь. Иногда компиляторы могут самостоятельно оптимизировать одинокостоящие операции i++, заменяя их на ++i, чтобы избежать дополнительных расходов на копирование.

1. В чём ключевое отличие лицензии на свободное ПО GPL от лицензии LGPL?

**Варианты ответов**:

1. GPL-программу можно скомпилировать и затем запустить в виде самостоятельного процесса изнутри закрытой коммерческой программы, тогда как LGPL этого не позволяет.
2. Лицензию GPL невозможно переделать в лицензию закрытого (коммерческого) типа, тогда как LGPL это позволяет, например, при использовании динамических библиотек.
3. При использовании LGPL требуется указывать авторство всех внесённых правок в исходный код, тогда как в GPL это относится только к случаю совместного использования кода в закрытом коммерческом приложении.
4. Исходный код GPL нельзя использовать в связке с закрытым коммерческим кодом, тогда как LGPL-код можно прилинковать к закрытому коду в виде библиотеки.

**Пояснения.** Исходный код GPL нельзя ни в каком виде использовать в связке с закрытым коммерческим кодом, тогда как исходный код LGPL позволяет сделать из него библиотеку и подключать эту библиотеку к коммерческому закрытому ПО (при этом при модификации кода LGPL-библиотеки внесённые изменения должны быть также лицензированы по LGPL).