Задания заключительного этапа
Всероссийской олимпиады студентов «Я – профессионал»
по направлению **«Программирование и информационные технологии»**

Категория участия «Бакалавриат»

Задание 1, 10 баллов

Ниже представлен скрипт, создающий несколько таблиц базы данных, которая хранит информацию ERP-системы:

```
CREATE TABLE IF NOT EXISTS "Orders"
(
    "OrderID"        INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    "OrderDate"      DATETIME,
    "RequiredDate"   DATETIME,
    "ShippedDate"    DATETIME
);

CREATE TABLE IF NOT EXISTS "Products"
(
    "ProductID"       INTEGER             NOT NULL PRIMARY KEY
AUTOINCREMENT,
    "ProductName"     TEXT                NOT NULL,
    "UnitPrice"       NUMERIC DEFAULT 0,
    "UnitsInStock"    INTEGER DEFAULT 0,
    "UnitsOnOrder"    INTEGER DEFAULT 0,
    CHECK ("UnitPrice" >= (0)),
    CHECK ("UnitsInStock" >= (0)),
    CHECK ("UnitsOnOrder" >= (0))
);

CREATE TABLE IF NOT EXISTS "OrderDetails"
(
    "OrderID"   INTEGER           NOT NULL REFERENCES "Orders",
    "ProductID" INTEGER           NOT NULL REFERENCES "Products",
    "UnitPrice" NUMERIC DEFAULT 0 NOT NULL,
    "Quantity"  INTEGER DEFAULT 1 NOT NULL,
    PRIMARY KEY ("OrderID", "ProductID"),
    CHECK ("Quantity" > (0)),
    CHECK ("UnitPrice" >= (0))
);
```

Стажёр отдела бизнес-аналитики успешно прошёл испытательный срок, и теперь ему нужно выполнить следующую задачу: для каждого продукта определить, на какую сумму (нарастающим итогом) было всего продано этого продукта после каждой продажи. При

этом известно, что при добавлении записей в таблицы Orders и Products первичные ключи всегда генерировались автоматически.

Для решения задачи сотрудник написал следующий запрос:

```
SELECT
    "ProductID",
    "OrderID",
    (
        SELECT SUM("UnitPrice" * "Quantity")
        FROM "OrderDetails"
        WHERE "ProductID" = OD."ProductID" and "OrderID" <=
OD."OrderID"
    ) "RunningTotal"
FROM "OrderDetails" OD
ORDER BY "ProductID", "OrderID";
```

Оказалось, что запрос не может быть выполнен за приемлемое для клиента время, и его нужно переписать. Напишите запрос, который выполняет ту же задачу, что и представленный выше.

В качестве ответа введите один SQL-запрос, который будет использовать ровно один оператор SELECT. Ваш запрос должен возвращать такую же выборку, как и запрос в условии, с учётом порядка строк и столбцов.

У вас есть 5 попыток на отправку ответа к заданию. Запрос с синтаксическими ошибками будет засчитан как неправильный ответ и использует попытку, поэтому рекомендуем проверить корректность запроса на локальном компьютере до отправки. При подсчёте баллов будет учитываться только последний отправленный ответ.

Примечание: для анализа данных сотрудник использует СУБД SQLite, но гарантируется, что приведённый выше запрос выполнится корректно в СУБД SQLite, PostgreSQL и MS SQL Server. При необходимости вместо кавычек при указании полей можно использовать квадратные скобки.

Ответ:

```
SELECT
    "ProductID",
    "OrderID",
    SUM("UnitPrice" * "Quantity") OVER (PARTITION BY "ProductID" ORDER
 BY "OrderID") "RunningTotal"
FROM "OrderDetails"
ORDER BY "ProductID", "OrderID";
```

Задание 2, 9 баллов

Одним из способов организации памяти вычислительной системы является выделение памяти процессам фиксированными разделами различного размера. Такой подход, в частности, используется для встроенных систем, когда количество процессов и их потребности в памяти являются относительно стабильными и предсказуемыми. В этом случае можно существенно выиграть по накладным расходам на вычисление физических адресов, но при этом для надежной работы системы требуется аккуратное определение параметров. Вам требуется произвести оценку конфигурации для заданной модели, чтобы определить, возможна ли аварийная ситуация при определенных параметрах и если да, то с каким процессом она произойдет.

Описание модели

Время системы дискретно и измеряется в условных тактах, начальный такт имеет номер 0.

В системе рождаются, исполняются и завершаются процессы трех типов. Для каждого из типов процесса известна периодичность рождения процессов этого типа (экземпляр процесса рождается в такт, номер которого кратен заданному параметру K, в начальный такт рождаются процессы всех трех типов), длительность исполнения (количество тактов L, за которое процесс исполнится и освободит память) и минимальный объем памяти в КБайт, требуемый для размещения процесса, M.

Тип процесса	K	L	M
P1	2	16	2
P2	4	8	4
P3	8	4	8

Для того, чтобы начать исполняться, процесс должен быть размещен в подходящем разделе оперативной памяти. Всего для процессов доступны 7 разделов памяти: 4 раздела по 2 КБайта, 2 раздела по 4 КБайта и 1 раздел в 8 КБайт.

Диспетчер процессов работает по следующему алгоритму.

1. В начале каждого такта рождаются все процессы, которые должны родиться в этот такт. Каждому рождающемуся процессу последовательно присваивается идентификатор (PID) – очередное целое неотрицательное число. Первому родившемуся процессу присваивается PID, равный 0. Если в один такт рождаются процессы разных типов, то PID присваиваются по возрастанию номера типа процесса. Родившийся процесс помещается в конце очереди процессов к разделу памяти минимально возможного для его размещения

размера, независимо от наличия или отсутствия в этот момент свободных разделов памяти того или иного размера.

2. Освобождаются разделы памяти, используемые теми процессами, которые завершили исполнение в конце предыдущего такта. PID завершившихся процессов не может быть повторно назначен новому процессу.
3. Поочередно по возрастанию размеров разделов рассматриваются очереди процессов к разделам соответствующего размера. Если соответствующая очередь не пуста и есть свободные разделы памяти этого размера, процессы из начала очереди размещаются в разделах и в этом такте начинают исполняться, исключаясь из очереди.

Временем на выполнение пунктов 1-3 пренебрегаем, считая, что эти операции исполнились мгновенно.

4. В течение очередного такта исполняются все процессы, которые находятся в состоянии исполнения. На момент окончания такта фиксируется время ожидания всех процессов в очередях (очереди просматриваются по возрастанию размеров разделов памяти). Если процесс ожидает 10 тактов в очереди к разделам размером 2 КБайт или в очереди к разделам размером 4 КБайт, он перемещается в конец очереди к разделам следующего по возрастанию размера. Если процесс ожидает суммарно в очередях 40 тактов, инициируется аварийная ситуация: работа системы останавливается и фиксируется PID такого процесса.

Определите, может ли возникнуть аварийная ситуация при указанных данных. Если она не может возникнуть, укажите в ответе NULL. Если аварийная ситуация возникнет, укажите в качестве ответа PID процесса, вызвавшего аварийную ситуацию.

Ответ: 20

Задание 3, 7 баллов

Для передачи данных в IP-сетях на хосте-отправителе нужно предварительно установить соответствие между IP-адресом получателя и MAC-адресом получателя. Это необходимо из-за того, что IP-адрес получателя известен отправителю, и этот адрес может быть записан в поле адреса получателя IP-пакета. В то же время MAC-адрес заранее неизвестен, но необходим для корректной инкапсуляции IP-пакета в кадр канального уровня, где в поле адреса получателя тоже необходимо указать корректный MAC-адрес.

В IPv4 и IPv6 механизмы определения соответствия IP и MAC различаются.

В IPv4 работает протокол Address Resolution Protocol (ARP) по следующему алгоритму:

1. Хост, желающий отправить данные, сначала проверяет свою таблицу ARP, чтобы узнать, есть ли уже запись для IP-адреса назначения. Если такой записи нет, устройство отправляет широковещательный ARP-запрос на MAC-

адрес FF:FF:FF:FF:FF:FF. Это сообщение доставляется всем узлам сети без исключения, и все узлы обрабатывают его.

2. Если IP-адрес назначения не совпадает с собственным IP-адресом узла, запрос игнорируется.
3. Если IP-адрес назначения совпадает с собственным IP-адресом узла, узел отвечает со своего MAC-адреса на MAC-адрес узла, отправившего запрос. Это сообщение получает только исходный узел.
4. Исходный узел пополняет свой ARP-кэш записью о соответствии IP-адреса MAC-адресу.

Далее идет стандартная передача пакетов.

В IPv6 часто используют протокол ICMPv6 в режиме Neighbor Discovery Protocol (NDP). Он решает ту же задачу, что и ARP, но использует не широковещательные, а мультикаст-рассылки. Различие состоит в том, что мультикаст-сообщение получают только узлы одной группы, а не все узлы сети без разбору. В NDP используются следующие мультикаст-адреса:

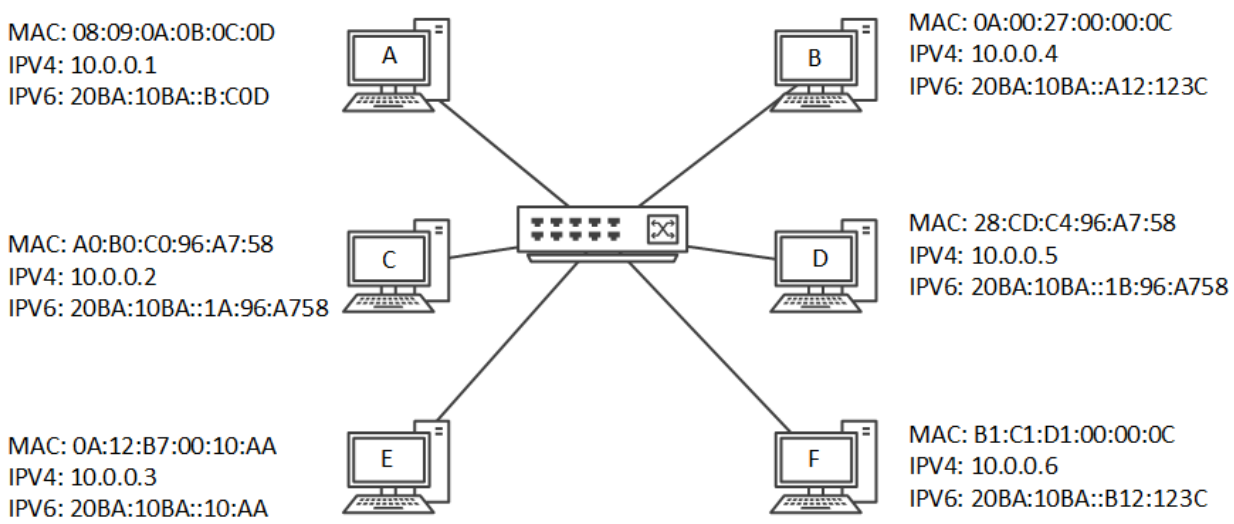
1. FF02::1 – все IPv6-узлы в сети.
2. SNMA-адреса (Solicited-node multicast address) – группа узлов, для которых IPv6-адрес заканчивается на одинаковые 24 бита. Этот адрес формируется добавлением к префиксу FF02::1:FF/104 последних трёх байт искомого IPv6-адреса. Например, для адреса 2001:DB8::1234:5678 соответствующий SNMA-адрес - FF02::1:FF34:5678.

Будем считать, что сам протокол NDP работает так:

1. Хост, желающий отправить данные, сначала проверяет свою таблицу ND, чтобы узнать, есть ли уже запись для искомого IPv6-адреса назначения. Если такой записи нет, устройство отправляет мультикаст-запрос на SNMA-адрес искомого IPv6-адреса. При этом в поле данных запроса содержится искомый IPv6-адрес. Такой запрос называется Neighbor Solicitation.
2. Это сообщение получают все узлы мультикаст-группы, то есть все узлы, у которых последние 3 байта IPv6-адреса совпадают. Тот узел, IPv6-адрес которого точно совпадет с искомым IPv6-адресом, отправляет ответ на IPv6-адрес узла, пославшего Neighbor Solicitation.
3. Исходный узел пополняет свой ND-кэш записью о соответствии IPv6-адреса MAC-адресу.

Далее идет стандартная передача пакетов.

На рисунке приведена схема сети. У каждого узла есть IPv4-адрес и IPv6-адрес. Узел A последовательно пополняет свою ARP-таблицу, потом ND-кэш, занося в них адреса всех остальных компьютеров в сети.



Определите, сколько сообщений получит узел D, пока узел A заносит адреса всех узлов в ARP-кэш и пока узел A заносит адреса всех узлов в ND-кэш.

В ответе укажите числа через запятую, сначала для ARP-кэша, потом для ND-кэша (например: 12,34).

Примечание: при записи IPv6-адреса не пишутся ведущие нули в 2-байтных группах и 2-байтные группы, содержавшие нули. Например,
адрес 2001:0DB8:00AF:ABCD:0000:0000:0000:0034 будет записываться
так: 2001:DB8:AF:ABCD::34.

Ответ: 5,2

Задание 4, 8 баллов

Пользователь написал некоторую программу на ассемблере для архитектуры x86, ориентируясь на 32-битную версию платформы. В своей программе пользователь использовал передачу данных в функции через стек, используя стековый регистр. В архитектуре x86 стековый регистр обозначается как esp (Extended Stack Pointer) и используется для управления стеком данных в процессоре. Стек представляет собой область памяти, работающую по принципу *Last In, First Out* (LIFO), это означает, что последний элемент, помещенный в стек, будет извлечен первым. esp указывает на вершину стека, т.е. на адрес в памяти, где находится текущий верхний элемент стека. При выполнении операций push (занесение значения на стек) и pop (извлечение значения со стека) регистр esp автоматически обновляется, чтобы указывать на новую вершину стека. При выполнении операции push в процессоре x86 значение esp уменьшается, указывая на новую вершину стека, а при выполнении операции pop значение esp увеличивается, указывая на предыдущую вершину стека. Стек в архитектуре компьютера используется для управления вызовами функций, передачи параметров, сохранения локальных переменных и возврата из функций.

Операции push и pop используются для добавления и удаления данных со стека соответственно. Когда функция вызывается с помощью инструкции call, адрес возврата (адрес следующей инструкции после call) помещается в стек. Затем в самой функции push ebp сохраняет текущее значение регистра ebp (базового указателя кадра стека, frame pointer), который будет использоваться для доступа к локальным переменным и параметрам функции. В конце функции pop ebp и ret используются для восстановления состояния стека и возврата из функции.

Дан текст программы:

```
section .data
section .text
    global _start


_start: ;точка входа


    mov eax, 10
    push eax
    call fun_one
    mov edx, eax


    ; завершение программы
    mov eax, 1        ; Код завершения программы
    xor ebx, ebx      ; Стандартный код завершения
    int 0x80          ; Вызываем системный вызов для завершения
программы


fun_one: ;метка функции fun_one
    ; Код функции fun_one
    push ebp
    mov ebp, esp
    mov eax, [ebp+8]
    add eax, 5
    push eax
    call fun_two
```

```
    mov esp, ebp
    pop ebp
    ret

fun_two: ;метка функции fun_two
    ; Код функции fun_two
    push ebp
    mov ebp, esp
    mov eax, [ebp+8]
    shl eax, 1
    mov esp, ebp
    pop ebp
    ret
```

Описание используемых команд:

- Команда add (от *add*) в ассемблере используется для выполнения операции сложения. Синтаксис команды обычно выглядит как add destination, source, где destination — место, куда будет добавлено значение, а source — значение, которое будет добавлено к destination. Результат сложения сохраняется в destination.
- Команда mov (от *move*) в ассемблере предназначена для перемещения (копирования) данных из одного места в другое. Синтаксис команды обычно выглядит как mov destination, source, где destination — место, куда будет скопировано значение, а source — значение, которое будет скопировано.
- Команда call в ассемблере используется для вызова процедуры или функции (подпрограммы). Синтаксис команды call обычно выглядит так: call target, где target — метка (или адрес), обозначающая начало процедуры, которую нужно вызвать.
- Команда shl (от *shift left*) в ассемблере x86 используется для выполнения логического сдвига бит влево. Синтаксис команды shl обычно выглядит так: shl destination, count, где destination — операнд, значение которого будет сдвигаться влево, и count — количество бит, на которое следует сдвинуть destination.
- Команда ret (от *return*) в ассемблере используется для завершения выполнения подпрограммы (функции) и возвращения из неё в вызывающий код. Синтаксис команды ret обычно выглядит так: ret.

Описание регистров:

- eax (*accumulator*): используется для хранения результатов операций и для передачи параметров в системные вызовы;
- ebx (*base*): обычно используется как базовый регистр для адресации в массивах и структурах данных;
- edx (*data*): используется в операциях деления и умножения, а также для хранения данных в различных операциях.

Символ «;» используется для обозначения начала строчного комментария.

Какое значение будет в регистре edx в момент окончания работы программы?

В ответе введите одно число в десятичной системе счисления.

Ответ: 30

Задание 5, 6 баллов

Задан набор точек на плоскости:

[1, 10], [17, -20], [-5, 6], [11, 15], [-5, 14], [-18, -6], [-1, -9], [-1, -4], [-14, 0], [13, -8], [13, -17]

[15, 18], [6, -9], [-14, 1], [-1, -3], [18, 7], [-5, 18], [13, 17], [-7, 7], [8, 19], [16, -15], [-16, 6],

[-12, -11], [-8, 4], [5, 11]

Алгоритм определения начальных центров (центроидов) для кластеризации k-средних.

1. В качестве первого центроида берем первую точку набора [1, 10].
2. В цикле по числу итераций $k = 1,2, \ldots$
   a. Для всех точек набора, не являющихся центроидами, вычисляем расстояния до ближайшего из ранее полученных центроидов. Расстояние между точками вычисляем в евклидовой метрике:

$$\rho\big((x_1, y_1), (x_2, y_2)\big) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

   b. В качестве очередного центроида (с номером $k + 1$) выбираем первую в наборе точку, имеющую максимальное расстояние до ближайшего к ней центроида. Само максимальное расстояние обозначим, как ρ_k. Соответственно, расстояние, вычисленное на первом шаге есть ρ_1.

Требуется: определить минимальное число центроидов, полученных за k итераций ($k+1$ центроид), необходимых, чтобы максимальное расстояние до ближайшего центроида ρ_k, удовлетворяло условию:

$$2\rho_k < \rho_1$$

В качестве ответа введите минимальное число центроидов.

Ответ: 6

Задание 6, 9 баллов

Граф называется вершинным кактусом, если никакая вершина не лежит на одновременно более чем на одном простом цикле. Какое минимальное число вершин может быть у вершинного кактуса, чтобы у него было 2024 различных остовных дерева?

Ответ: 40

Задание 7, 9 баллов

Для строки, состоящей из букв английского алфавита, вычислили z-функцию. Выяснилось, что `z[10]=4`, `z[12]=4`. Какие значения может принимать `z[11]`?

a. 0

b. 1

c. 2

d. 3

e. 4

f. 5

g. 6

h. такая ситуация невозможна

Ответ: 0, 4

Задание 8, 5 баллов

Ограничение времени	1 секунда
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Задано натуральное число n.

Требуется найти минимальное целое число $k \geq n$, такое что k делится на сумму своих цифр в десятичной системе счисления.

Формат ввода

На ввод подается натуральное число n ($1 \leq n \leq 109$).

Формат вывода

Выведите искомое число k.

Пример

Ввод	**Вывод**
13	18

Задание 9, 8 баллов

Ограничение времени	1 секунда
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Заданы целые числа L и R, $L \leq R$.

Требуется выбрать такие два числа a и b, чтобы выполнялись неравенства $L \leq a \leq R$, $L \leq b \leq R$, $b \neq 0$ и вещественное число a / b было как можно меньше.

Формат ввода

В этой задаче входные данные содержат несколько тестовых примеров.

На первой строке ввода находится число t — количество тестовых примеров ($1 \le t \le 1000$).

На каждой из t следующих строк содержится два числа L и R ($-109 \le L \le R \le 109$, не бывает ситуации L=R=0).

Формат вывода

Выведите t строк. Для каждого тестового примера выведите два искомых целых числа a и b. Если подходящих ответов несколько, выведите любой из них.

Пример

Ввод	**Вывод**
2 5 10 -3 3	5 10 3 -1

Задание 10, 7 баллов

Ограничение времени	1 секунда
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Будем называть строку из букв «a» и «b» хорошей, если в ней нет трех одинаковых символов подряд.

Вам дана строка s из букв «a» и «b». Какое минимальное число символов в ней необходимо изменить, чтобы она стала хорошей?

Формат ввода

В этой задаче входные данные содержат несколько тестовых примеров.

На первой строке ввода находится число t — количество тестовых примеров.

На каждой из t следующих строк содержится один тест — непустая строка s, состоящая из английских букв «a» и «b».

Суммарная длина строк во всех тестовых примерах одних входных данных не превышает 200000.

Формат вывода

Выведите t чисел. Для каждого тестового примера выведите одно число — минимальное количество символов, которое необходимо изменить в строке s, чтобы она стала хорошей.

Пример

Ввод	**Вывод**
3 abab aaaa bbbaa	0 1 1

Задание 11, 9 баллов

Ограничение времени	1 секунда
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Рассмотрим целые числа от 1 до n. Рассмотрим разбиение этих чисел на несколько множеств. Каждое число должно принадлежать ровно одному множеству.

Будем называть разбиение *интересным*, если ни в каком множестве нет трех (не обязательно различных) чисел x, y и z, таких что x+y=z.

Пусть, например, n=3. Разбиение [{1,2},{3}] не является интересным, так как 1+1=2. В то же время, разбиения [{1,3},{2}], [{1},{2,3}] и [{1},{2},{3}] являются интересными. Легко видеть, что только эти три разбиения являются интересными для n=3 (оставшееся разбиение [{1,2,3}] также не является интересным, так как 1+1=2, а также 1+2=3).

Найдите количество интересных разбиений на множества целых чисел от 1 до n.

Формат ввода

На ввод подается одно целое число n (1≤n≤15).

Формат вывода

Выведите одно число: количество интересных разбиений на множества целых чисел от 1 до n.

Пример

Ввод	**Вывод**
3	3

Задание 12, 13 баллов

Ограничение времени	5 секунд
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Рассмотрим неориентированный связный граф G. Для каждой вершины u определим как c(u) количество пар вершин (x,y), таких что любой путь из x в y проходит через вершину u.

Для каждой вершины u найдите значение c(u).

Формат ввода

В первой строке находятся два целых числа n и m — количество вершин и ребер в графе, соответственно (2≤n≤200000, 1≤m≤200000).

Следующие m строк содержат по два целых числа — номера вершин, соединенных ребрами. Гарантируется, что граф связен, никакие две вершины не соединены более чем одним ребром, никакое ребро не соединяет вершину саму с собой.

Формат вывода

Выведите n целых чисел: c(1),c(2),…,c(n).

Пример

Ввод	**Вывод**
8 9	7
1 3	7
1 4	7
3 4	17
4 6	7
6 7	19
6 8	13
7 8	13
8 2	
7 5	