Открытая олимпиада школьников (информатика) 2024-25
Задания заключительного этапа для 10 и 11 класса

1. Кодирование информации. Системы счисления (2 балла) [100 первых цифр]

Петя составляет ряд из периодических дробей, записанных в шестнадцатеричной системе счисления. Первый член ряда равен $0,(AB)_{16}$. Каждый следующий член ряда ровно в 4 раза меньше предыдущего. Известно, что у некоторого члена ряда сумма первых 100 цифр после запятой равна 441_{10}. Найдите и укажите в ответ номер этого члена ряда. Если такого члена ряда не существует, запишите в ответ NULL.

2. Кодирование информации. Объем информации (2 балла) [Полёт]

Петя создает ПО для управления дроном. Дрон перемещается в трехмерном пространстве с заданной декартовой системой координат. Дрон умеет выполнять 6 возможных команд: вверх, вниз, вправо, влево, вперед и назад. Каждая команда перемещает дрон на 1 единицу длины в указанном направлении. Пространство не содержит никаких препятствий для перемещения дрона. Программа для дрона – это последовательность из определенного количества указанных команд. Пока дрон умеет выполнять только программы, состоящие ровно из 11 команд. Петя решил записывать программы для дрона в память в виде последовательности команд, кодируя каждую команду минимально возможным, одинаковым для всех команд количеством бит.

Вася заметил, что дрон всегда стартует из точки A и всегда заканчивает движение в точке B, смещенную на 2 единицы длины вправо, 3 единицы длины вверх и 4 единицы длины вперед. Вася сказал Пете, что тогда нет смысла выделять память исходя из необходимости хранить произвольную программу заданной длины. Он предложил перенумеровать натуральными числами все программы, которые смогут переместить дрона из точки A в точку B и сохранять в памяти номер такой программы, используя для каждого возможного номера минимально возможное, одинаковое для всех номеров количество бит. Насколько меньше бит потребуется Васе для хранения одной программы, чем предлагал Петя? В ответе укажите целое число.

3. Основы логики (3 балла) [Кто такой Жегалкин?]

Есть логический преобразователь, на вход которому подается логическая функция от 20 переменных, не являющаяся тождественной ложью (то есть имеющая значение ИСТИНА хотя бы для одного набора значений переменных). Работает преобразователь следующим образом:

1. Строится таблица истинности данной функции.
2. Из полученной таблицы истинности берутся наборы переменных, для которых значение функции равно ИСТИНЕ.
3. Для каждого такого набора составляется полная конъюнкция всех 20 переменных, причем если значение этой переменной в наборе равно ЛЖИ, то в конъюнкции эта переменная будет с отрицанием, если значение переменной равно ИСТИНЕ, то в конъюнкции эта переменная будет без изменений.
4. Между всеми полученными полными конъюнкциями ставятся операции $\oplus$ (Исключающее ИЛИ).
5. Все переменные с отрицанием заменяются по следующему правилу: $\bar{X} = (X \oplus 1)$
6. Раскрываются все скобки по правилу: $X \wedge (Y \oplus Z) = X \wedge Y \oplus X \wedge Z$. После раскрытия скобок должно получиться выражение, которое может содержать некоторые отдельные переменные, конъюнкции переменных и константу ИСТИНА, соединенные Исключающим ИЛИ.
7. Из выражения убираются повторяющиеся конъюнкции. В результате у нас получается новая форма записи исходной функции. Гарантируется, что для каждой входной функции эта форма уникальна.
8. Производится проверка, остались ли в получившейся форме записи выражения операции конъюнкции. Если остались – преобразователь возвращает ЛОЖЬ, если нет – возвращает ИСТИНА.

Сколько существует таких функций от 20 переменных, которые можно подать на вход преобразователю и для которых он вернет ИСТИНУ? В ответе укажите целое число.

4. Кодирование информации. Алгоритмы обработки кодированной информации (1 балл) [Разбегайся!]

Пусть N – шестизначное десятичное число. Последовательность цифр R получается по следующему алгоритму:

1. 1000 раз повторяется первая (самая левая) цифра числа N.
2. Цикл для всех последующих цифр числа N, двигаясь слева направо:
   После каждого элемента последовательности, полученной на предыдущем шаге цикла вставить два раза подряд эту цифру

Например, если N=123456, то первые 10 элементов последовательности R, получившейся после завершения алгоритма будут:

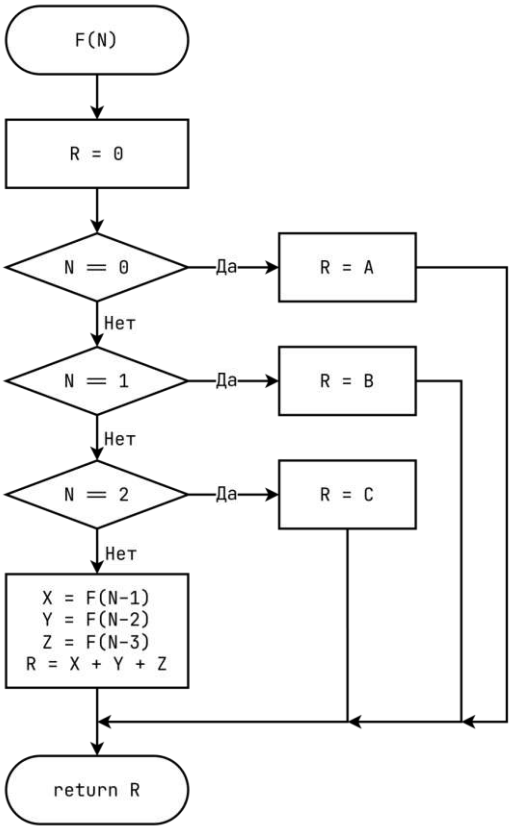
1, 6, 6, 5, 6, 6, 5, 6, 6, 4

Известно, что в получившейся последовательности элемент с номером 209161 равен 7, элемент с номером 242758 равен 3, а элемент с номером 189890 равен 1. Нумерация элементов слева направо с 1. Определите максимальное число N, при котором это возможно, и запишите его в ответ. Если такое число невозможно, запишите в ответ NULL.

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (1 балл)

[Непизанские кролики]

Дана блок-схема алгоритма, реализованного в виде рекурсивно вызываемой функции:



Известно, что A, B и C - натуральные числа и что $A < B < C$.

Петя вызывает эту функцию, передавая ей в качестве входного параметра натуральное число. Известно, что для некоторого k функция возвращает следующие значения:

$F(k) = 50890368413$;

$F(k+1) = 93601980590$;

$F(k+2) = 172160883161$.

Определите значения A, B и C, при которых это возможно. Если таких вариантов несколько, выберите вариант с наименьшей суммой A, B и C. В ответе введите в указанном порядке значения A, B и C, разделённые пробелом.

6. Телекоммуникационные технологии (2 балла).

[BGP]

Интернет состоит из отдельных крупных сетей, управляемых отдельными организациями. Между такими сетями распределяется пространство IP-адресов и устанавливается техническое и организационное взаимодействие. Несколько упрощая, можно сказать, что такие крупные сети называются автономными системами (AS). За каждой AS закрепляется IP-подсеть.

Для обмена маршрутной информацией между маршрутизаторами AS используется внешний протокол динамической маршрутизации BGP (Border Gateway Protocol). BGP выбирает наилучший маршрут для передачи пакетов на основе ряда атрибутов и метрик.

BGP-роутер при маршрутизации пакета выбирает IP-адрес следующего по маршруту маршрутизатора, который в BGP называется «соседом» (neighbor), по набору атрибутов маршрутов. В задаче для упрощения будем считать, что реализуется следующий алгоритм.

1. по адресу назначения в IP-пакете выбираются подходящие маршруты из таблицы маршрутизации по полю «сеть назначения».
2. потом из них выбирается лучший, для чего **последовательно** проверяются атрибуты маршрутов. То есть сначала проверяется первый атрибут всех подходящих маршрутов, и если у возможных маршрутов он имеет равное значение, проверяется второй атрибут и т.д.

В задаче используются следующие атрибуты в указанном порядке:

1. Локальное предпочтение (Local Preference)

Local Preference — это атрибут, который используется для выбора исходящего трафика из автономной системы. Маршрут с **наибольшим** значением Local Preference считается предпочтительным.

2. Наименьший Путь AS (AS Path)

BGP предпочитает маршруты с **наименьшим** количеством автономных систем в пути (AS Path). Чем короче путь, тем лучше.

3. Наименьший IGP Metric до Next Hop

Если есть несколько маршрутов с одинаковыми атрибутами, BGP выбирает маршрут с **наименьшим** значением метрики IGP до Next Hop (следующего прыжка).

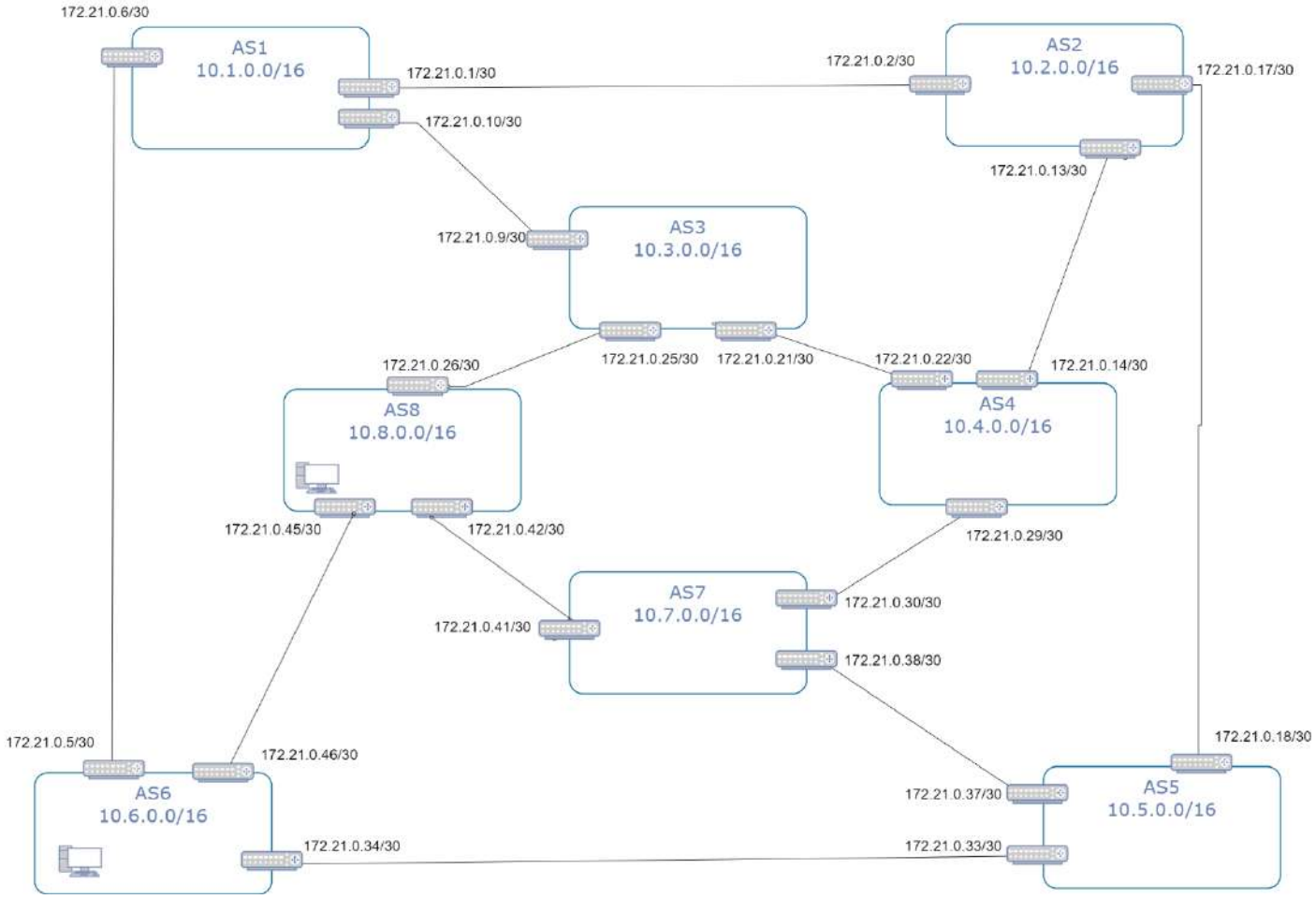
4. Старейший маршрут (Oldest Route)

Если все предыдущие атрибуты равны, BGP предпочитает **старейший** маршрут, так как он считается более стабильным.

5. Наименьший IP-адрес соседа

В крайнем случае, если все предыдущие атрибуты равны, BGP выбирает маршрут, полученный от соседа с **наименьшим** IP-адресом (сравнение производится как сравнение 32-битных значений адресов).

Далее приведены схема сети и фрагменты таблиц маршрутов для каждой из AS (сделано это для простоты, в реальности эти данные есть в каждом граничном маршрутизаторе AS).



AS1:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.2
10.1.0.0/16	Локально	500	0	0	2025-02-01	-
10.3.0.0/16	AS3	200	1	10	2025-02-03	172.21.0.9
10.7.0.0/16	AS2	100	3	20	2025-02-03	172.21.0.2
10.8.0.0/16	AS3	100	2	20	2025-02-01	172.21.0.9
10.6.0.0/16	AS6	100	2	20	2025-02-01	172.21.0.5
10.6.0.0/16	AS3	100	2	20	2025-02-01	172.21.0.9
10.3.0.0/16	AS6	200	3	50	2025-02-23	172.21.0.5
10.4.0.0/16	AS3	200	2	5	2025-02-03	172.21.0.9

AS2:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.2.0.0/16	Локально	500	0	0	2025-02-01	-
10.3.0.0/16	AS1	100	2	20	2025-02-03	172.21.0.1
10.6.0.0/16	AS1	100	2	20	2025-02-10	172.21.0.1
10.7.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.14
10.8.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.6.0.0/16	AS5	100	2	20	2025-02-03	172.21.0.18
10.3.0.0/16	AS4	100	2	20	2025-02-03	172.21.0.14
10.4.0.0/16	AS1	200	2	20	2025-02-03	172.21.0.1
10.4.0.0/16	AS4	100	1	20	2025-02-03	172.21.0.14
10.1.0.0/16	AS1	100	1	20	2025-02-03	172.21.0.1
10.1.0.0/16	AS4	200	3	20	2025-02-03	172.21.0.14

AS3:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	2	20	2025-02-02	172.21.0.22
10.6.0.0/16	AS1	100	2	20	2025-02-01	172.21.0.10
10.6.0.0/16	AS4	200	4	10	2025-02-03	172.21.0.22
10.7.0.0/16	AS4	100	3	10	2025-02-04	172.21.0.22
10.3.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS8	100	3	10	2025-02-04	172.21.0.26
10.4.0.0/16	AS4	200	1	10	2025-02-04	172.21.0.22
10.4.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS8	100	3	20	2025-02-04	172.21.0.26
10.1.0.0/16	AS1	100	1	20	2025-02-04	172.21.0.10

AS4:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.13
10.6.0.0/16	AS2	100	3	5	2025-02-03	172.21.0.13
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	AS7	100	3	10	2025-02-01	172.21.0.30
10.8.0.0/16	AS3	100	1	10	2025-02-01	172.21.0.21
10.6.0.0/16	AS3	100	3	5	2025-02-11	172.21.0.21
10.1.0.0/16	AS3	100	3	5	2025-02-01	172.21.0.21

AS5:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.4.0.0/16	AS2	100	2	10	2025-02-02	172.21.0.17
10.5.0.0/16	Локально	500	0	0	2025-02-01	-
10.6.0.0/16	AS6	100	1	20	2025-02-03	172.21.0.34
10.8.0.0/16	AS7	100	3	20	2025-02-01	172.21.0.38
10.3.0.0/16	AS2	500	3	20	2025-02-01	172.21.0.17
10.2.0.0/16	AS2	100	3	20	2025-02-01	172.21.0.17
10.3.0.0/16	AS6	100	3	20	2025-02-01	172.21.0.34
10.4.0.0/16	AS7	100	2	20	2025-02-02	172.21.0.38
10.1.0.0/16	AS6	100	2	20	2025-02-02	172.21.0.34
10.1.0.0/16	AS2	100	2	20	2025-02-02	172.21.0.17

AS6:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS1	100	3	20	2025-02-02	172.21.0.6
10.6.0.0/16	Локально	500	0	0	2025-02-01	-
10.4.0.0/16	AS5	100	3	51	2025-02-11	172.21.0.33
10.7.0.0/16	AS1	100	4	20	2025-02-03	172.21.0.6
10.8.0.0/16	AS1	100	3	20	2025-02-01	172.21.0.6
10.3.0.0/16	AS8	200	2	5	2025-03-08	172.21.0.45
10.4.0.0/16	AS1	100	3	52	2025-02-11	172.21.0.6
10.3.0.0/16	AS5	100	4	50	2025-02-11	172.21.0.33
10.4.0.0/16	AS8	100	3	50	2025-02-11	172.21.0.45

AS7:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.5.0.0/16	AS4	100	3	20	2025-02-02	172.21.0.29
10.6.0.0/16	AS5	100	2	20	2025-02-03	172.21.0.37
10.7.0.0/16	Локально	500	0	0	2025-02-01	-
10.8.0.0/16	AS4	100	3	20	2025-02-01	172.21.0.29
10.4.0.0/16	AS4	100	1	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS8	300	3	20	2025-02-01	172.21.0.42
10.1.0.0/16	AS4	200	3	20	2025-02-01	172.21.0.29
10.1.0.0/16	AS5	400	5	80	2025-02-01	172.21.0.37

AS8:

Сеть назначения	Номер следующей AS	Local Preference	Длина AS Path	IGP Metric до Next Hop	Дата появления маршрута	Адрес соседа
10.6.0.0/16	AS3	200	3	10	2025-02-01	172.21.0.25
10.7.0.0/16	AS3	200	3	10	2025-02-05	172.21.0.25
10.5.0.0/16	AS3	200	2	10	2025-02-03	172.21.0.25
10.4.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.8.0.0/16	Локально	500	0	0	2025-02-01	-
10.2.0.0/16	AS7	200	2	10	2025-02-03	172.21.0.41
10.6.0.0/16	AS7	200	3	20	2025-02-01	172.21.0.41
10.3.0.0/16	AS3	200	1	10	2025-02-15	172.21.0.25
10.4.0.0/16	AS3	100	2	10	2025-02-03	172.21.0.25
10.6.0.0/16	AS6	50	1	20	2025-02-03	172.21.0.46

Определите сначала маршрут прохождения пакета из AS8 в AS6, а потом маршрут прохождения пакета в случае, если из-за аварии будет отключены связи между AS8 и AS3, а также между AS1 и AS6, а записи, соответствующие этим связям, будут исключены из таблиц.

В ответ запишите номера автономных систем через запятую, начиная с 8 и заканчивая 6, сначала для первого случая, а потом, через тире, для второго. Если маршрут построить нельзя (например, его нет или образуется цикл), вместо перечисления номеров автономных систем укажите NULL.

Пример ввода ответа: 8,10,12,6-8,10,1,2,6.

7. Технологии сортировки и фильтрации данных (2 балл)
[Космический капитализм]

В свободное от работы время Николай Владимирович, ответственный за систему проведения олимпиад, разрабатывает новый космический симулятор, и на данный момент у него готова система инвентаря корабля с оборудованием, гиперпрыжки между двумя звёздными системами и торговля на планетах. Рассмотрим подробнее устройство симулятора.

Игровой мир состоит из звёздных систем, в каждой из которых есть несколько планет. Для перемещения корабля между звёздными системами необходим двигатель и топливный бак. На один гиперпрыжок корабль тратит количество топлива, равное расстоянию между звёздными системами в парсеках. Из этого следует, что корабль не может выполнить прыжок на расстояние больше, чем вместимость его топливного бака.

Владелец корабля имеет определённое количество галактических кредитов на счёте. На каждой планете представлены товары пяти видов: продовольствие, медикаменты, техника, роскошь и минералы. Для каждой планеты и каждого вида товара известно количество имеющегося товара в тоннах, стоимость покупки и продажи в галактических кредитах. Корпус корабля имеет заданную грузоподъёмность, и нельзя купить товаров больше, чем имеющаяся грузоподъёмность.

Сейчас Николай Владимирович пишет тест для проверки этой функциональности. Тестовый сценарий состоит из нескольких шагов:

1. Приобрести товар **одного вида** на текущей планете в некотором количестве;
2. Взлететь с планеты и выполнить гиперпрыжок **в другую систему**;
3. Приземлиться на выбранной планете и продать весь купленный на шаге 1 товар.

Для этого сценария есть отдельный файл сохранения, для которого известно количество товара на каждой планете, стоимость его покупки и продажи. Эти данные представлены в виде отдельной базы данных, которая доступна ниже.

Таблица stars хранит информацию о звёздных системах:

- id — уникальный идентификатор системы в рамках игрового мира;
- name — название системы.

Таблица planets хранит информацию о планетах:

- id — уникальный идентификатор планеты в рамках игрового мира;
- name — название планеты;
- star_id — идентификатор звёздной системы, к которой относится планета.

Таблица star_map хранит информацию о расстояниях между звёздными системами:

- star_from_id — идентификатор начальной звёздной системы;
- star_to_id — идентификатор звёздной системы назначения (обязательно отличается от star_from_id);
- distance — расстояние между этими звёздными системами в парсеках.

Таблица product_types хранит информацию о типах товаров:

- id — уникальный идентификатор типа товаров;
- name — название типа товаров.

Таблица prices хранит информацию о стоимости покупки и продажи каждого типа товаров на каждой из планет:

- planet_id — идентификатор планеты;
- product_type — идентификатор типа товаров;
- buy_price — стоимость покупки владельцем корабля одной тонны товара в галактических кредитах;
- sell_price — стоимость продажи владельцем корабля одной тонны товара в галактических кредитах;
- amount — количество товара в тоннах, доступное к покупке на данной планете.

В рамках данного сценария кредитов хватит для покупки любого количества любого товара на любой планете. Пока что в этом сценарии не определено, какой товар и в каком количестве нужно приобрести и на какой планете его продать. Николай Владимирович хочет подобрать эти параметры так, чтобы максимизировать полученную прибыль (разницу между стоимостью продажи и покупки). Известно, что в сохранении корабль находится на планете Раалито системы Хезе, грузоподъёмность корабля равна 80 тонн, а вместимости топливного бака хватит на прыжок расстоянием в 25 парсек. Как опытный программист, Николай Владимирович определил нужные параметры за несколько минут и дописал тест. Определите это вместе с ним и вы. В качестве ответа введите одно целое число — прибыль в галактических кредитах, полученную после выполнения тестового сценария с найденными параметрами. Если окажется, что для тестового сценария нет ни одного набора параметров, который бы дал положительную прибыль, в качестве ответа введите 0.

Задание 8 (3 балла) и Задание 9 (4 балла). Технологии программирования

В отдельном архиве опубликованы тесты, решения жюри и проверяющие программы для задач по технологиям программирования. Архив содержит 4 директории: по два варианта каждой из двух задач. В поддиректории tests каждой задачи содержатся тесты и ответы решения жюри в файлах "??" и "??.a" соответственно, где "??" -- двузначное число с, возможно, ведущим нулем. В поддиректории solutions каждой задачи содержатся решения жюри на различных языках программирования, а также примеры неверных решений (см. файлы с расширением *.desc, чтобы классифицировать решения). В корневом директории архива находится программа, использовавшаяся для проверки ответа участника (check.cpp). В поддиректории statements содержатся условия задач. Апелляции заданий по программированию должны быть основаны на предоставленных тестах.

Наименования директорий с задачами (по вариантам): задача 8 – jobs-scheduling-v1, jobs-scheduling-v2; задача 9 – tournament-rewards-v1, tournament-rewards-v2.

Задача 11A1. Расписание станков

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Известный завод ждет реорганизация — его собираются переоборудовать n новейшими станками. Перед тем, как эти станки будут установлены, требуется разработать интерфейс для обработки задач на этих станках.

После реорганизации наладчик завода будет распределять поступающие задачи между станками. У каждой задачи есть длительность. У каждого станка есть независимая очередь задач, причем новую задачу можно добавить либо строго в конец этой очереди, либо строго в начало, если это *срочная задача.*

Станки работают строго в порядке очереди, обрабатывая задачи подряд. Как только станок завершает задачу, он сразу же начинает работу над следующей в очереди, если такая есть. На переключение между задачами время не тратится.

Формально определим время начала работы над задачей как первый момент времени, после которого прогресс по задаче строго увеличится. Так, если в момент времени t на свободный станок приходят сначала обычная задача A и затем срочная задача U, временем начала U станет t, а временем начала A — момент завершения U.

Аналогично, время завершения работы над задачей — первый момент времени, когда прогресс по задаче достигает ее длительности. Так, если в момент времени t прогресс по задаче A достиг ее длительности, и станку поступил запрос на обработку срочной задачи U, временем завершения A все равно будет t.

Для большего понимания советуем после прочтения условия ознакомиться с иллюстрацией внизу.

Всего поддерживается четыре типа запросов.

1. Добавить задачу номер i длительностью d в конец очереди k-го станка. Если его очередь до этого была пустой, станок тут же начинает работу над добавленной задачей.

2. Отменить задачу с номером i. Если эта задача еще не была начата, она удаляется из очереди, а порядок остальных задач в очереди не меняется.

   Если эта задача уже была начата, работа над ней тут же останавливается, и станок берет в работу следующую задачу из очереди. Если эта задача уже была завершена или такой задачи не было, запрос отмены игнорируется.

3. Добавить *срочную задачу* с номером i длительностью d в начало очереди k-го станка. В этом случае работа над текущей задачей (если она есть) на k-м станке приостанавливается с сохранением прогресса, и в работу берется данная срочная задача.

   До момента завершения срочной задачи все запросы добавления или отмены новых задач к станку номер k **игнорируются** для экономии ресурсов. Иными словами, если в момент времени t поступил запрос добавления срочной задачи, все следующие запросы, поступающие к k-му станку в моменты времени с t **включительно** до $t + d$ **не включительно** будут проигнорированы.

   По завершении срочной задачи, если работа над какой-то обычной задачей в очереди была приостановлена, эта задача без задержек возвращается в обработку.

4. Вывести ожидаемый момент времени завершения работы k-го станка.

   Для станка без задач в очереди «ожидаемым» временем завершения его работы будем считать текущий момент времени.

Еще раз **обратите внимание**, что во время исполнения срочной задачи станок игнорирует только все запросы добавления и отмены задач (включая добавление другой срочной задачи). Про

запрос отмены будем считать, что он идет к тому станку, в очереди которого лежит соответствующая задача. Запросы вывода ожидаемого момента завершения работы **не игнорируются**.

Вам необходимо помочь наладчику и вывести для каждой задачи время начала и завершения работы над ней.

Формат входных данных

В первой строке дано целое число T ($1 \leqslant T \leqslant 1000$) — количество наборов входных данных.

В первой строке описания набора входных данных через пробел даны два целых числа n и q ($1 \leqslant n, q \leqslant 10^5$) — количество станков и количество запросов. Гарантируется, что сумма n и сумма q по всем наборам входных данных обе не превосходят 10^5.

В следующих q строках описаны запросы к заводу в одном из следующих форматов:

1. «t `schedule` i `(`d`) on` k» — добавить задачу на k-й станок;
2. «t `cancel` i» — отменить задачу;
3. «t `urgent` i `(`d`) on` k» — добавить срочную задачу на k-м станке;
4. «t `expectation` k» — узнать текущее ожидаемое время завершения работы k-го станка.

Параметр t — момент совершения запроса (целое неотрицательное число от 0 до 10^9). Для всех запросов выполняется $1 \leqslant k \leqslant n$, $1 \leqslant i \leqslant 10^9$ и $1 \leqslant d \leqslant 10^9$.

Также гарантируется, что номера задач (i) уникальны и не повторяются, а запросы упорядочены по времени, то есть t каждого следующего запроса не меньше, чем t предыдущего.

Формат выходных данных

Выведите в отдельной строке текущее ожидаемое время работы соответствующего станка после каждого запроса типа «`expectation`».

Затем выведите по строке на каждую задачу, запрос на добавление которой не был проигнорирован. Задачи должны быть упорядочены по возрастанию их номеров. В каждой строке через пробел должны быть выведены три числа: номер задачи, время начала ее обработки и время конца ее обработки (или отмены, если задача была отменена до завершения).

Для задач, которые были отменены до начала работы над ними, считайте время начала равным времени первого запроса их отмены.

Пример

стандартный ввод	стандартный вывод
3 2 2 0 schedule 1 (10) on 1 4 schedule 2 (3) on 2 3 6 0 schedule 1 (5) on 1 0 schedule 2 (5) on 2 2 urgent 3 (5) on 1 6 expectation 1 6 expectation 2 6 expectation 3 2 17 0 schedule 1 (10) on 1 0 schedule 2 (6) on 2 2 schedule 3 (5) on 1 3 schedule 4 (100) on 1 3 schedule 5 (1) on 2 5 cancel 3 5 urgent 6 (6) on 2 8 cancel 6 9 cancel 6 10 cancel 6 10 schedule 7 (150) on 2 10 urgent 8 (3) on 1 11 schedule 9 (3) on 2 14 expectation 1 14 cancel 4 14 schedule 10 (2) on 1 14 expectation 1	1 0 10 2 4 7 10 6 6 1 0 10 2 0 5 3 2 7 113 16 1 0 10 2 0 12 3 5 5 4 13 14 5 12 13 6 5 11 8 10 13 9 13 16 10 14 16

Замечание

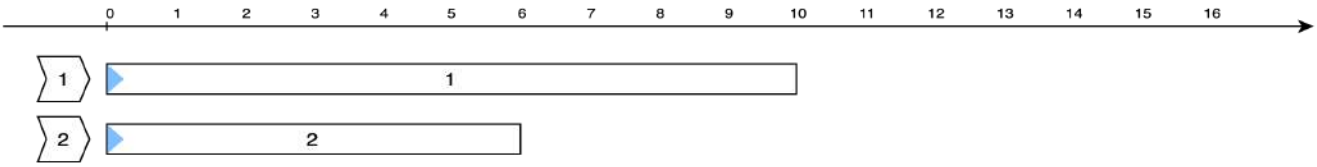
В первом примере из условия обе задачи будут обработаны по расписанию: с 0 до 10 и с 4 до 7 соответственно.

Во втором примере:

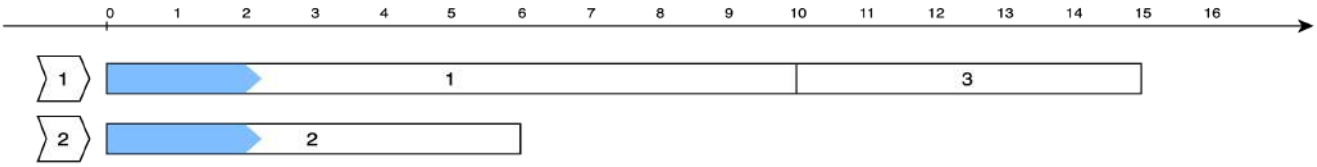
1. первая задача должна обрабатываться на первом станке в период $[0, 5]$;
2. вторая задача должна обрабатываться на втором станке в период $[0, 5]$;
3. в момент времени 2 срочная задача номер 3 добавляется на первый станок; она будет обрабатываться в период $[2, 7]$;
4. в момент времени 6 первому станку нужно еще 4 единицы времени на завершение задач 4 и 1; второй станок завершил работу в момент времени 5, а третий станок вообще не начинал работу — для них время ожидания до завершения работы равно 0.

Подробная иллюстрация к третьему примеру приведена ниже. Здесь синим обозначен прогресс по задачам, зеленым — завершенные задачи, красным — отмененные, а градиентом — срочные. Все интервалы короче 1 единицы времени (см. отмененную задачу 3) стоит воспринимать как интервалы длительностью 0.

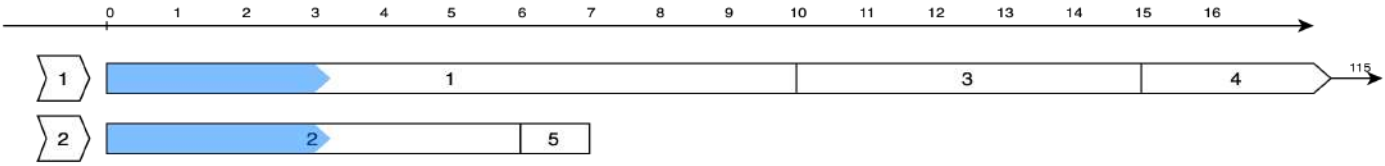
```
0 schedule 1 (10) on 1
0 schedule 2 (6) on 2
```



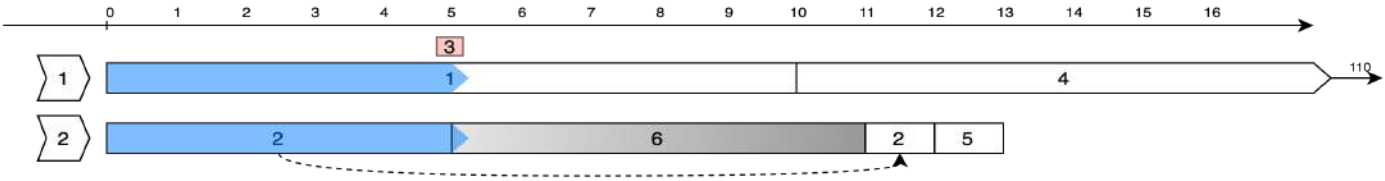

```
2 schedule 3 (5) on 1
```




```
3 schedule 4 (100) on 1
3 schedule 5 (1) on 2
```

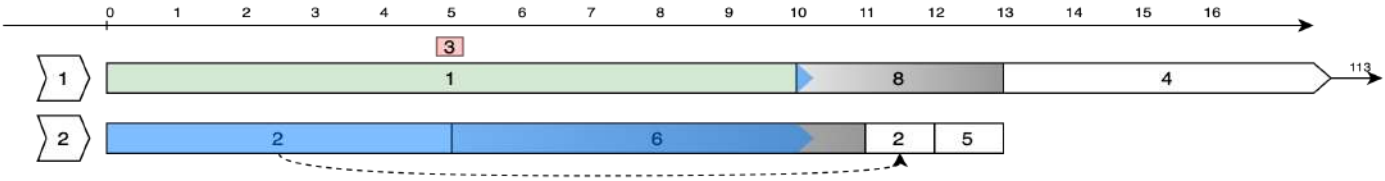



```
5 cancel 3
5 urgent 6 (6) on 2
```

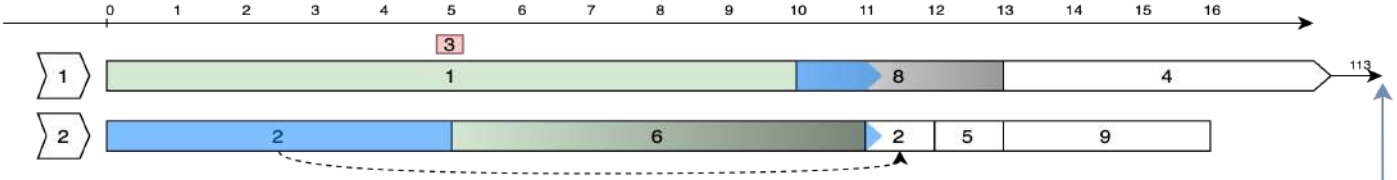



```
8 cancel 6
9 cancel 6
10 cancel 6
10 schedule 7 (150) on 2
10 urgent 8 (3) on 1
```

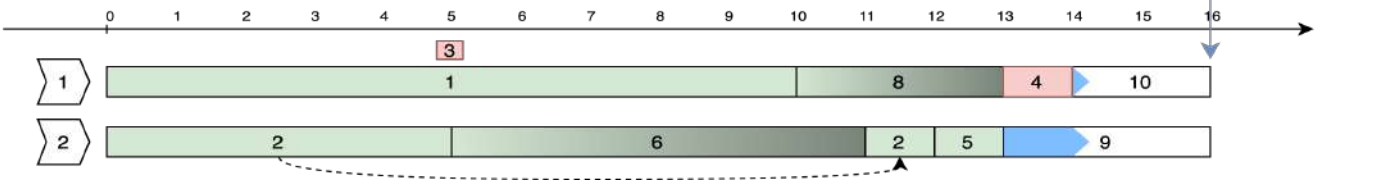
игнорируются, так как второй станок занят срочной задачей




```
11 schedule 9 (3) on 2
```




```
14 expectation 1
14 cancel 4
14 schedule 10 (2) on 1
14 expectation 1
```



Задача 11A2. Расписание станков

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Известный завод ждет реорганизация — его собираются переоборудовать n новейшими станками. Перед тем, как эти станки будут установлены, требуется разработать интерфейс для обработки задач на этих станках.

После реорганизации наладчик завода будет распределять поступающие задачи между станками. У каждой задачи есть длительность. У каждого станка есть независимая очередь задач, причем новую задачу можно добавить либо строго в конец этой очереди, либо строго в начало, если это *срочная задача.*

Станки работают строго в порядке очереди, обрабатывая задачи подряд. Как только станок завершает задачу, он сразу же начинает работу над следующей в очереди, если такая есть. На переключение между задачами время не тратится.

Формально определим время начала работы над задачей как первый момент времени, в который задача станет первой в очереди. Так, если в момент времени t на свободный станок приходят сначала обычная задача A и затем срочная задача U, обе эти задачи будут иметь время начала t.

Аналогично, время завершения работы над задачей — первый момент времени, когда прогресс по задаче достигает ее длительности. Так, если в момент времени t прогресс по задаче A достиг ее длительности, и станку поступил запрос на обработку срочной задачи U, временем завершения A все равно будет t.

Для большего понимания советуем после прочтения условия ознакомиться с иллюстрацией внизу.

Всего поддерживается четыре типа запросов.

1. Добавить задачу номер i длительностью d в конец очереди k-го станка. Если его очередь до этого была пустой, станок тут же начинает работу над добавленной задачей.

2. Отменить задачу с номером i. Если эта задача еще не была начата, она удаляется из очереди, а порядок остальных задач в очереди не меняется.

   Если эта задача уже была начата, работа над ней тут же останавливается, и станок берет в работу следующую задачу из очереди. Если эта задача уже была завершена или такой задачи не было, запрос отмены игнорируется.

3. Добавить *срочную задачу* с номером i длительностью d в начало очереди k-го станка. В этом случае работа над текущей задачей (если она есть) на k-м станке приостанавливается с сохранением прогресса, и в работу берется данная срочная задача.

   До момента завершения срочной задачи все запросы добавления или отмены новых задач к станку номер k **игнорируются** для экономии ресурсов. Иными словами, если в момент времени t поступил запрос добавления срочной задачи, все следующие запросы, поступающие к k-му станку в моменты времени с t **включительно** до $t + d$ **не включительно** будут проигнорированы.

   По завершении срочной задачи, если работа над какой-то обычной задачей в очереди была приостановлена, эта задача без задержек возвращается в обработку.

4. Вывести ожидаемое время до того, как k-й станок завершит работу над всеми задачами в своей очереди.

   Для станка без задач в очереди «ожидаемым» временем до завершения его работы будем считать 0.

Еще раз **обратите внимание**, что во время исполнения срочной задачи станок игнорирует только все запросы добавления и отмены задач (включая добавление другой срочной задачи). Запросы вывода ожидаемого времени до завершения работы **не игнорируются**.

Вам необходимо помочь наладчику и вывести для каждой задачи время начала и завершения работы над ней.

Формат входных данных

В первой строке дано целое число T ($1 \leqslant T \leqslant 1000$) — количество наборов входных данных.

В первой строке описания набора входных данных через пробел даны два целых числа n и q ($1 \leqslant n, q \leqslant 10^5$) — количество станков и количество запросов. Гарантируется, что сумма n и сумма q по всем наборам входных данных обе не превосходят 10^5.

В следующих q строках описаны запросы к заводу в одном из следующих форматов:

1. «t `:` `add` i`@`d `on` k» — добавить задачу на k-й станок;
2. «t `:` `cancel` i» — отменить задачу;
3. «t `:` `add-urgent` i`@`d `on` k» — добавить срочную задачу на k-м станке;
4. «t `:` `expected-time` k» — узнать текущее ожидаемое время до завершения работы k-го станка.

Параметр t — момент совершения запроса (целое неотрицательное число от 0 до 10^9). Для всех запросов выполняется $1 \leqslant k \leqslant n$, $1 \leqslant x \leqslant q$, $1 \leqslant i \leqslant 10^9$ и $1 \leqslant d \leqslant 10^9$.

Также гарантируется, что номера задач (i) уникальны и не повторяются, а запросы упорядочены по времени, то есть t каждого следующего запроса не меньше, чем t предыдущего.

Формат выходных данных

Выведите в отдельной строке текущее ожидаемое время до завершения работы соответствующего станка после каждого запроса типа «`expected-time`».

Затем выведите по строке на каждую задачу, запрос на добавление которой не был проигнорирован. Задачи должны быть упорядочены по возрастанию их номеров. Каждую строку выведите в формате «i s`->`f», где i — номер задачи, s — время начала ее обработки, и f — время конца ее обработки (или отмены, если задача была отменена до завершения).

Для задач, которые были отменены до начала работы над ними, считайте время начала равным времени первого запроса их отмены.

Пример

стандартный ввод	стандартный вывод
3 2 2 0 : add 1@10 on 1 4 : add 2@3 on 2 3 6 0 : add 1@5 on 1 0 : add 2@5 on 2 2 : add-urgent 3@5 on 1 6 : expected-time 1 6 : expected-time 2 6 : expected-time 3 2 17 0 : add 1@10 on 1 0 : add 2@6 on 2 2 : add 3@5 on 1 3 : add 4@100 on 1 3 : add 5@1 on 2 5 : cancel 3 5 : add-urgent 6@6 on 2 8 : cancel 6 9 : cancel 6 10 : cancel 6 10 : add 7@150 on 2 10 : add-urgent 8@3 on 1 11 : add 9@3 on 2 14 : expected-time 1 14 : cancel 4 14 : add 10@2 on 1 14 : expected-time 1	1 0->10 2 4->7 4 0 0 1 0->10 2 0->5 3 2->7 99 2 1 0->10 2 0->12 3 5->5 4 10->14 5 12->13 6 5->11 8 10->13 9 13->16 10 14->16

Замечание

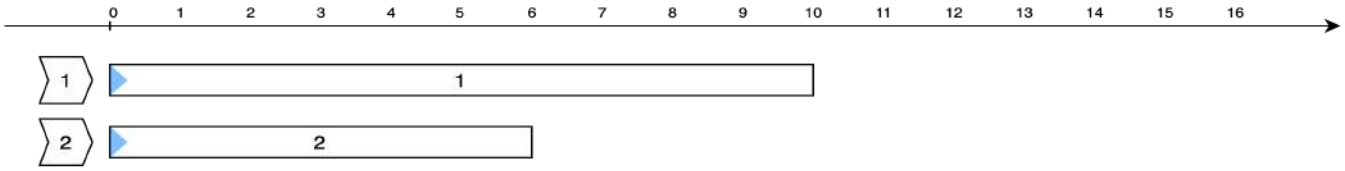
В первом примере из условия обе задачи будут обработаны по расписанию: с 0 до 10 и с 4 до 7 соответственно.

Во втором примере:

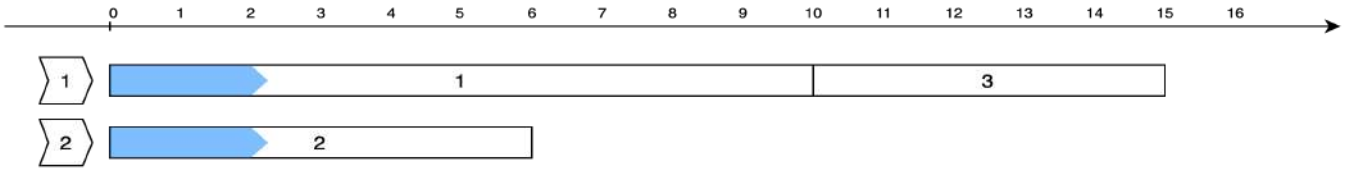
1. первая задача должна обрабатываться на первом станке в период $[0, 5]$;
2. вторая задача должна обрабатываться на втором станке в период $[0, 5]$;
3. в момент времени 2 срочная задача номер 3 добавляется на первый станок; она будет обрабатываться в период $[2, 7]$;
4. в момент времени 6 первому станку нужно еще 4 единицы времени на завершение задач 4 и 1; второй станок завершил работу в момент времени 5; третий станок вообще не начинал работу — его время завершения равно 0.

Подробная иллюстрация к третьему примеру приведена ниже. Здесь синим обозначен прогресс по задачам, зеленым — завершенные задачи, красным — отмененные, а градиентом — срочные. Все интервалы короче 1 единицы времени (см. отмененную задачу 3 и момент начала задачи 4) стоит воспринимать как интервалы длительностью 0.

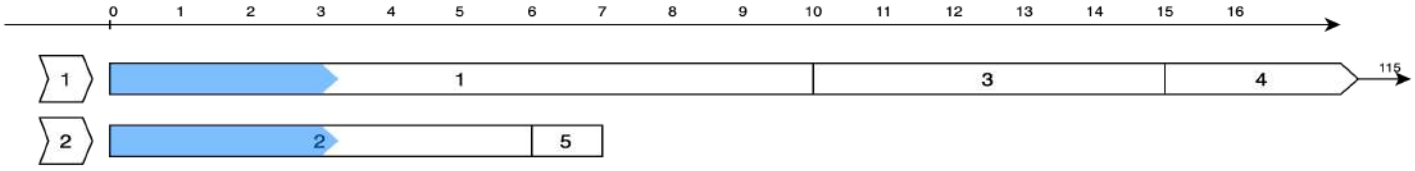
```
0 : add 1@10 on 1
0 : add 2@6 on 2
```



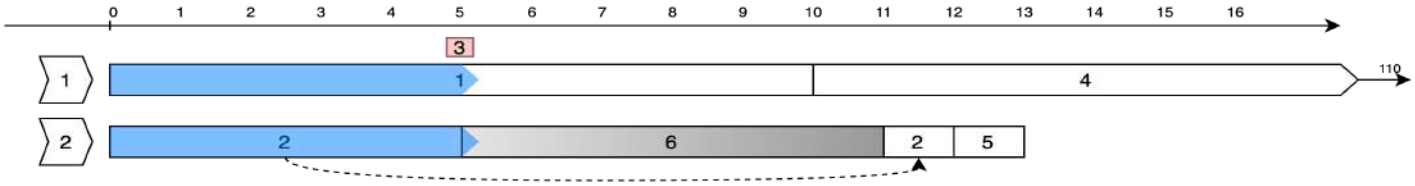

```
2 : add 3@5 on 1
```




```
3 : add 4@100 on 1
3 : add 5@1 on 2
```

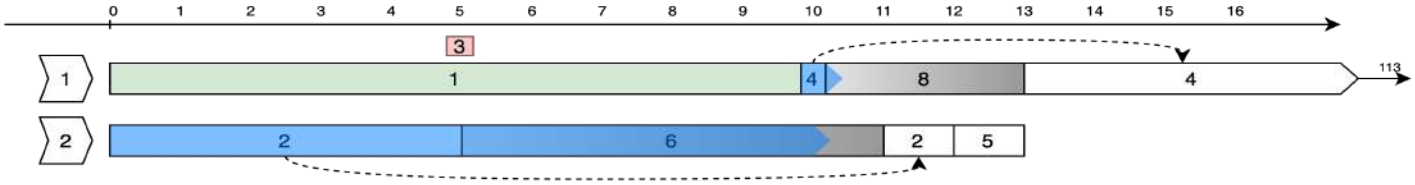



```
5 : cancel 3
5 : add-urgent 6@6 on 2
```

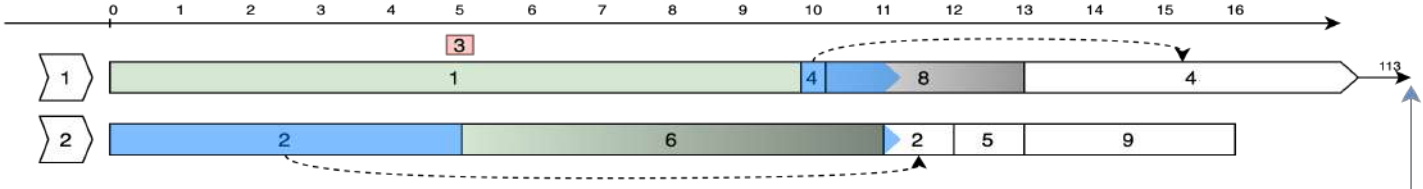



```
8 : cancel 6
9 : cancel 6
10 : cancel 6
10 : add 7@150 on 2
10 : add-urgent 8@3 on 1
```

игнорируются, так как второй станок занят срочной задачей




```
11 : add 9@3 on 2
```

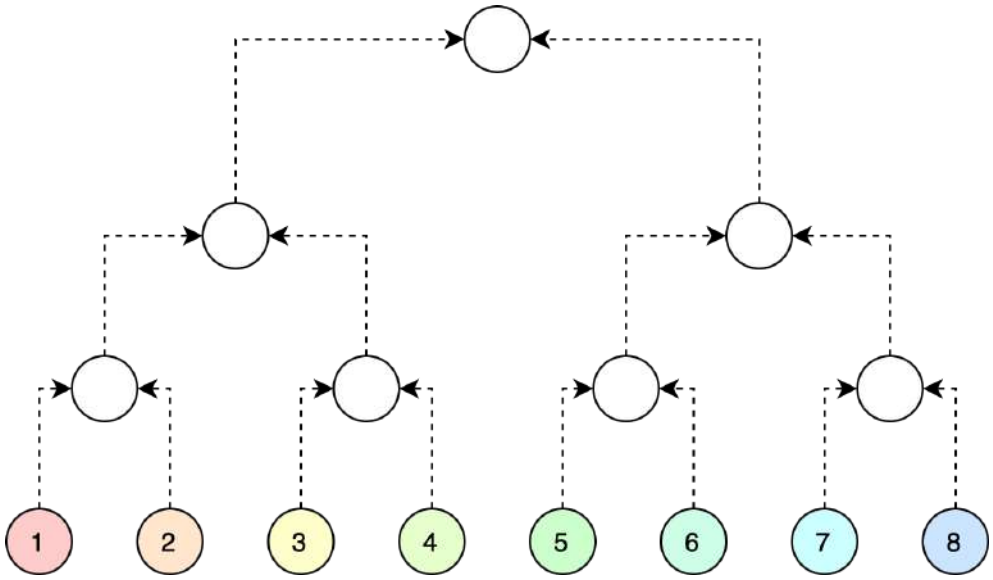



```
14 : expected-time 1
14 : cancel 4
14 : add 10@2 on 1
14 : expected-time 1
```

Задача 11B1. Бинарная Сила

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

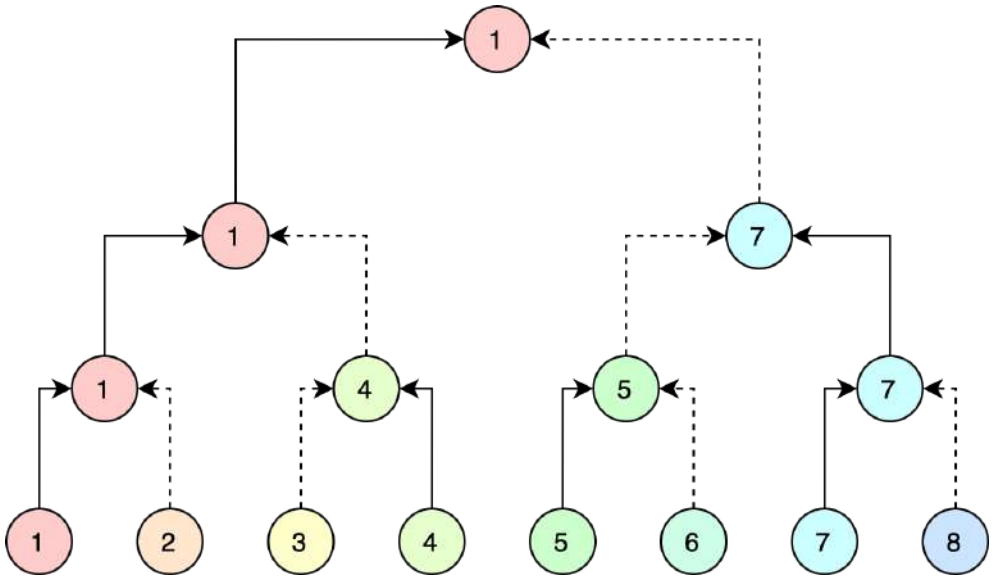
Вы играете в игру «Бинарная Сила» и управляете персонажем, у которого есть $d = 2^n$ навыков, пронумерованных от 1 до d. Эти навыки расположены на листьях полного двоичного дерева высоты n, изначально все навыки имеют уровень 1. Пример такого дерева для $n = 3$ приведен на иллюстрации ниже.



После этого вы начинаете прокачивать навыки следующим образом.

- Навыки прокачиваются посредством заполнения двоичного дерева снизу вверх.
- Для очередной вершины дерева вы должны выбрать и записать в нее один из двух навыков, записанных в непосредственных детях этой вершины (на рисунке из детей в родителя ведут стрелки).
- *Уровнем* навыка считается число вершин, в которых выбран этот навык.

Пример корректного распределения навыков по дереву для $n = 3$ приведен ниже.



В этом примере первый навык имеет уровень 4, седьмой — уровень 3, четвертый и пятый — уровень 2, а второй, третий, шестой и восьмой не были прокачаны ни разу, поэтому остались на уровне 1.

Кроме прокачки персонажа, в игре есть m различных квестов, с помощью которых можно получать монетки. Квесты активируются после того, как были все дерево навыков было заполнено.

Квесты бывают трех типов:

1. «`less` x_i k_i s_i» — вы получите s_i монет, если уровень навыка с номером x_i окажется **строго меньше** k_i.
2. «`exact` x_i k_i s_i» — вы получите s_i монет, если уровень навыка с номером x_i окажется **равен** k_i.
3. «`more` x_i k_i s_i» — вы получите s_i монет, если уровень навыка с номером x_i окажется **строго больше** k_i.

Так как монеты — очень ценный ресурс в игре «Бинарная Сила», вы хотите узнать максимальное количество монет, которое возможно получить с помощью имеющихся квестов после улучшения всех навыков.

Формат входных данных

Каждый тест состоит из нескольких независимых наборов входных данных. Первая строка содержит одно целое число t — количество наборов входных данных ($1 \leqslant t \leqslant 10^4$). Далее следует описание наборов входных данных.

Каждый набор начинается со строки, содержащей два целых числа n и m — высоту дерева навыков и количество квестов соответственно ($1 \leqslant n \leqslant 15$; $0 \leqslant m \leqslant 50\,000$). Число навыков при этом равно $d = 2^n$.

Далее следуют m строк, i-я из которых содержит четыре целых числа t_i, x_i, k_i, s_i — тип квеста и его описание ($1 \leqslant t_i \leqslant 3$; $1 \leqslant x_i \leqslant d$; $1 \leqslant k_i \leqslant n$; $1 \leqslant s_i \leqslant 10^9$). Типы квестов следуют в том же порядке, в котором они перечислены в условии: $t_i = 1$ соответствует квесту типа «`less`», $t_i = 2$ — квесту типа «`exact`» и $t_i = 3$ — квесту типа «`more`».

Гарантируется, что сумма d по всем наборам входных данных не превосходит 2^{15} и сумма m по всем наборам входных данных не превосходит $50\,000$.

Формат выходных данных

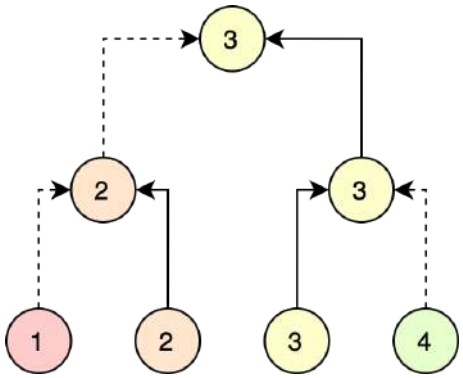
Для каждого набора выходных данных в отдельной строке выведите единственное число — максимальное количество монет, которые можно заработать.

Пример

стандартный ввод	стандартный вывод
4 2 5 2 1 1 10 2 2 2 100 1 2 2 5 3 3 1 15 2 3 1 10 1 3 2 2 1 5 2 1 1 10 3 1 1 3 10 6 2 1 1 1 2 2 2 1 2 4 3 1 2 8 4 1 2 16 5 1 2 32 6 1 10 4 1 6 3 5 1 6 4 7 2 6 5 11 3 6 1 3	125 10 6 15

Замечание

В первом примере из условия оптимальное распределение навыков выглядит следующим образом:



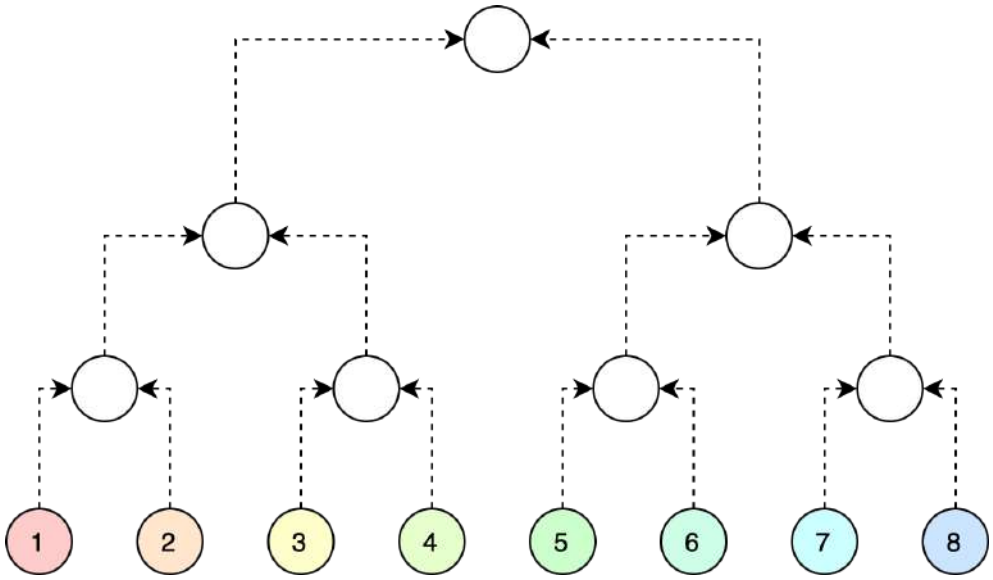
Действительно, мы получаем 100 монет за то, что второй навык имеет уровень **ровно** два, заодно получаем еще 10 монет за то, что первый имеет уровень **ровно** один. В этом случае третий квест не может быть выполнен, а из двух оставшихся мы получаем наибольшую награду (15 монет), если третий навык имеет уровень больше 1.

В третьем примере из условия достаточно для каждой вершины выбрать навык из правого ребенка. Тогда уровень навыка с номером 2^i будет в точности равен $i + 1$, и все квесты будут выполнены.

Задача 11B2. Бинарная Сила

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

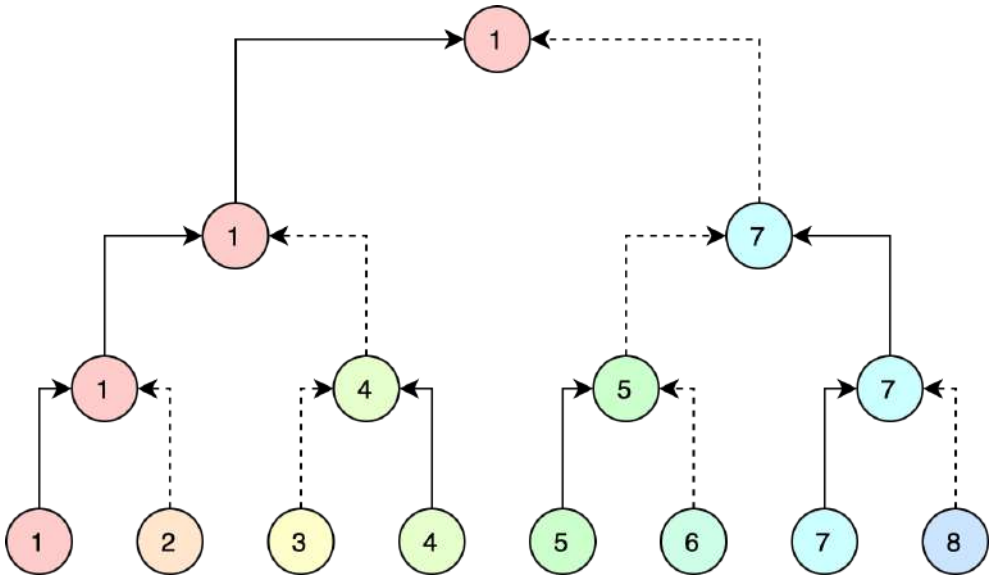
Вы играете в игру «Бинарная Сила» и управляете персонажем, у которого есть $d = 2^n$ навыков, пронумерованных от 1 до d. Эти навыки расположены на листьях полного двоичного дерева высоты n, изначально все навыки имеют уровень 1. Пример такого дерева для $n = 3$ приведен на иллюстрации ниже.



После этого вы начинаете прокачивать навыки следующим образом.

- Навыки прокачиваются посредством заполнения двоичного дерева снизу вверх.
- Для очередной вершины дерева вы должны выбрать и записать в нее один из двух навыков, записанных в непосредственных детях этой вершины (на рисунке из детей в родителя ведут стрелки).
- *Уровнем* навыка считается число не-листовых вершин (исходные d листьев не считаются), в которых выбран этот навык.

Пример корректного распределения навыков по дереву для $n = 3$ приведен ниже.



В этом примере первый навык имеет уровень 3, седьмой — уровень 2, четвертый и пятый — уровень 1, а второй, третий, шестой и восьмой не были прокачаны ни разу, поэтому остались на уровне 0.

Кроме прокачки персонажа, в игре есть m различных квестов, для прохождения которых может быть необходимо потратить монеты. Квесты активируются после того, как были все дерево навыков было заполнено.

Квесты бывают трех типов:

1. «`leq` s_i x_i k_i» — если уровень навыка с номером x_i окажется **меньше либо равен** k_i, вам придется потратить s_i монет.

2. «`neq` s_i x_i k_i» — если уровень навыка с номером x_i окажется **не равен** k_i, вам придется потратить s_i монет.

3. «`geq` s_i x_i k_i» — если уровень навыка с номером x_i окажется **больше либо равен** k_i, вам придется потратить s_i монет.

Так как монеты — очень ценный ресурс в игре «Бинарная Сила», вы хотите узнать минимальное количество монет, которое понадобится потратить для прохождения всех m квестов после улучшения всех навыков.

Формат входных данных

Каждый тест состоит из нескольких независимых наборов входных данных. Первая строка содержит одно целое число t — количество наборов входных данных ($1 \leqslant t \leqslant 10^4$). Далее следует описание наборов входных данных.

Каждый набор начинается со строки, содержащей два целых числа n и m — высоту дерева навыков и количество квестов соответственно ($1 \leqslant n \leqslant 15$; $0 \leqslant m \leqslant 50\,000$). Число навыков при этом равно $d = 2^n$.

Далее следуют m строк, i-я из которых содержит символ p_i и три целых числа s_i, x_i, k_i — тип квеста и его описание ($p_i \in \{$‘`v`’, ‘`x`’, ‘`^`’$\}$; $1 \leqslant s_i \leqslant 10^9$; $1 \leqslant x_i \leqslant d$; $0 \leqslant k_i < n$). Типы квестов следуют в том же порядке, в котором они перечислены в условии: ‘`v`’ соответствует квесту типа «`leq`», ‘`x`’ — квесту типа «`neq`» и ‘`^`’ — квесту типа «`geq`».

Гарантируется, что сумма d по всем наборам входных данных не превосходит 2^{15} и сумма m по всем наборам входных данных не превосходит $50\,000$.

Формат выходных данных

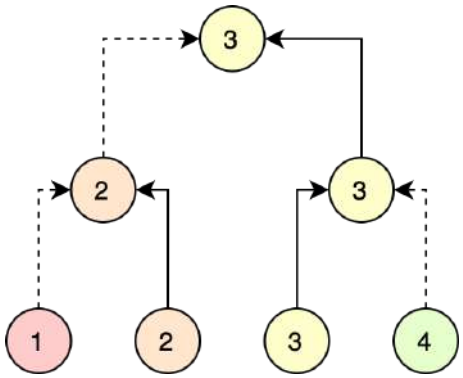
Для каждого набора выходных данных в отдельной строке выведите единственное число — минимальное количество монет, которое придется потратить.

Пример

стандартный ввод	стандартный вывод
4 2 5 x 10 1 0 x 100 2 1 ^ 5 1 0 v 15 3 0 x 10 3 0 1 3 x 5 2 0 x 10 1 0 v 3 1 0 10 6 x 1 1 0 x 1 2 1 x 1 4 2 x 1 8 3 x 1 16 4 x 1 32 5 10 4 ^ 5 6 2 ^ 7 6 3 x 11 6 4 v 3 6 0	15 8 0 11

Замечание

В первом примере из условия оптимальное распределение навыков выглядит следующим образом:



Действительно, мы бы потратили 100 монет, если бы второй навык имел уровень **не равный** одному. Делая его уровень равным 1, заодно бесплатно проходим первый квест, так как первый навык имеет уровень 0. Но в этом случае третий квест не может быть выполнен, а из двух оставшихся мы платим меньше (10 монет), если третий навык имеет уровень, не равный 0.

В третьем примере из условия достаточно для каждой вершины выбрать навык из правого ребенка. Тогда уровень навыка с номером 2^i будет в точности равен $i + 1$, и все квесты будут пройдены бесплатно.