Болгарский и русский языки — родственные, и многие болгарские слова похожи на русские. Например, болгарское слово «дърво» похоже на русское слово «дерево» и действительно переводится как «дерево». Но похожие русские и болгарские слова необязательно обозначают одно и то же. Поэтому для решения задачи нужно провести исследование всех слов и рассуждение.

Команды-запросы *Робота*. Условие

Представьте себе начальника, который отдаёт приказы, но не получает никаких донесений о результатах их выполнения; повара, который не может попробовать блюдо; шофёра, ведущего автомобиль с закрытыми глазами. Понятно, что так далеко не уедешь: если мы хотим составлять алгоритмы для решения сложных задач, то надо уметь не только командовать исполнителем, но и **анализировать обстановку**, в которой он оказался.

Например, для *Робота* **обстановка** — это поле, на котором он находится (расположение стен на поле, раскраска клеток, температура и уровень радиации в каждой клетке), а также то, в какой клетке находится *Робот*.

Для анализа обстановки у *Робота* есть 12 команд-запросов, будем для краткости называть их просто **запросами:**

Команды-запросы *Робота*

```
сверху стена
снизу стена
справа стена
слева стена
сверху свободно
снизу свободно
справа свободно
слева свободно
клетка закрашена
клетка чистая
температура
радиация
```

Отвечая на запрос, *Робот* сообщает информацию об обстановке в той клетке, где он находится: о закраске этой клетки, о наличии стен на границах клетки, о температуре и радиации в клетке.

Смысл этих вопросов ясен из их имён. Мы будем использовать пока только первые 10 команд-запросов — с командами-запросами `температура` и `радиация` мы будем работать позже.

Итак, рассмотрим команды-запросы, которые касаются наличия или отсутствия стены на границах клетки и того, закрашена ли клетка, на которой стоит *Робот*. Когда *Робот* получает такую команду, он определяет истинность соответствующего утверждения и даёт ответ — «истина» или «ложь». Можно считать, что, давая такую команду-запрос, компьютер или человек задаёт *Роботу* вопрос и получает от него в ответ «да» или «нет».

В нижней части пульта дистанционного управления *Роботом* (см. рисунок слева) есть кнопки «Стена/Закрашено», «Свободно/Чисто». Например, чтобы дать *Роботу* команду-запрос `сверху стена`, надо нажать на пульте две кнопки: сначала кнопку «Стена», а потом кнопку со стрелкой вверх. Чтобы дать *Роботу* команду-запрос `клетка чистая`, надо нажать на пульте кнопку «Чисто», а потом центральную кнопку. Для получения ответов *Робота* используется экран пульта — на нём появляется слово «да» («истина») или «нет» («ложь»).

Задача 1. *Робот* охраняет помещение, состоящее из двух соседних клеток (см. рисунок справа). Неизвестно, в какой из двух клеток находится *Робот*. Необходимо перевести его в другую клетку.

Сначала решим задачу в режиме непосредственного управления *Роботом*. По условию задачи мы не видим, в какой клетке поля он находится. Чтобы узнать, в какой клетке *Робот* сейчас находится, дадим ему команду-запрос: справа свободно. Ответ «истина» («да») означает, что *Робот* в левой клетке и должен сделать шаг вправо. Ответ «ложь» («нет») означает, что *Робот* в правой клетке и должен сдвинуться влево. Наши действия при непосредственном управлении *Роботом* можно описать так: *если* справа свободно, то скомандовать вправо, *иначе* (если справа не свободно, стоит стена) скомандовать влево.

При решении задачи 1 мы давали *Роботу* запрос и в зависимости от ответа выбирали следующее действие. Для описания такого выбора в алгоритмическом языке существует специальная *составная команда* **если**. Общий вид команды **если** таков:

```
если <условие>
  то
     <последовательность команд 1>
  иначе
     <последовательность команд 2>
все
```

или

```
если <условие>
  то
     <последовательность команд 1>
все
```

Служебные слова **если**, **то**, **иначе** имеют тот же смысл, что и в русском языке. Слово **все** означает конец команды («всё»), это слово пишется строго под словом **если**. После **то** записывается последовательность команд (последовательность команд 1), которая выполняется в случае истинности условия. После **иначе** записывается другая последовательность команд (последовательность команд 2), которая выполняется в случае ложности условия. Последовательность команд 2 вместе со служебным словом **иначе** может отсутствовать.

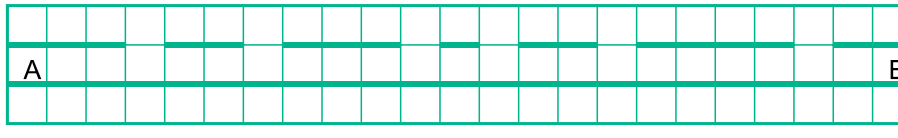
При обработке составной команды **если** компьютер посылает *Роботу* команду-запрос, чтобы проверить истинность условия,

записанного в строке **если**. Если *Робот* сообщает, что условие истинно, то компьютер посылает *Роботу* команды из последовательности 1, а команды из последовательности 2 пропускает. Если *Робот* сообщает, что условие ложно, то компьютер посылает *Роботу* команды из последовательности 2 (если она есть). Если условие ложно, а последовательность команд 2 вместе с **иначе** отсутствует, то компьютер сразу переходит к выполнению команд, записанных после слова **все**.

Вот как можно записать алгоритм решения задачи 1:

```
алг дежурство
  дано | Робот в одной из клеток
       | «двухкомнатного» помещения
  надо | Робот в другой клетке
нач
  если справа свободно
    то вправо
    иначе влево
  все
кон
```

Задача 2. *Робот* стоит в левом конце горизонтального коридора длиной в 23 клетки. Нижняя стена коридора сплошная, а в верхней имеется несколько выходов. Надо перевести *Робота* в правый конец коридора (из А в В) и закрасить все те клетки коридора, из которых есть выход вверх (см. рисунок). Количество выходов и их точное расположение заранее неизвестны.



Итак, *Робот* должен пройти по горизонтальному коридору, при этом закрашивать надо только те клетки коридора, где есть выход вверх. Другими словами, если сверху свободно, то клетку надо закрасить, иначе — красить не надо. Попробуем записать это в виде составной команды **если**:

```
если сверху свободно
  то закрасить
  иначе ...
все
```

А что собственно «иначе»? Как записать, что иначе ничего делать не надо? Для таких случаев и существует краткая форма составной команды **если**, в которой слова **иначе** вообще нет:

```
если сверху свободно
  то закрасить
все
```

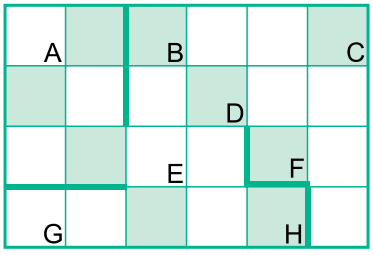
При выполнении этой составной команды компьютер даст *Роботу* команду определить истинность условия сверху свободно. Если *Робот* ответит «истинно», то компьютер скомандует *Роботу* закрасить. Если же *Робот* ответит «ложно», то компьютер вызов команды закрасить пропустит. Вот алгоритм решения задачи 2:

```
алг разметка выходов из коридора
  дано | Робот в левой клетке
       | горизонтального коридора
  надо | Робот в самой правой клетке коридора,
       | клетки коридора, из которых
       | есть выход вверх, закрашены
нач
  если сверху свободно
    то закрасить
  все
  нц 22 раз
    вправо
    если сверху свободно
      то закрасить
    все
  кц
кон
```

Обрати внимание: в этом алгоритме первая составная команда **если** служит для того, чтобы *Робот* раскрасил клетку А, если из неё есть выход из коридора. Дальше *Робот* 22 раза повторяет следующее: делает шаг вправо, проверяет, свободно ли сверху, и закрашивает клетку, если свободно. Если сверху стоит стена, *Робот* клетку не закрашивает, а сразу переходит на шаг вправо.

102

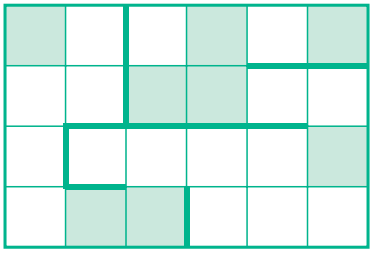
Определи истинность каждого утверждения (условия) в таблице для каждой из отмеченных клеток поля. Заполни такую таблицу в тетради — напиши в каждой клетке букву И или Л.



Утверждение \ Клетка	A	B	C	D	E	F	G	H
сверху свободно								
справа стена								
снизу свободно								
слева стена								
клетка чистая								

103

Пользуясь заполненной таблицей, для каждого имени найди клетку на поле, которой соответствуют указанные значения истинности утверждений (условий). Нарисуй такое поле в тетради, расставь имена клеток так, чтобы таблица была верной.

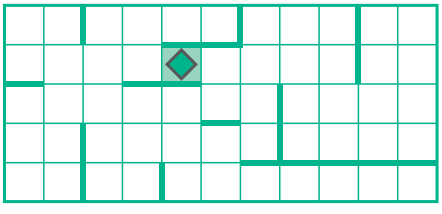


Утверждение \ Клетка	K	L	M	N	O	P	Q	R
сверху свободно	И	Л	Л	И	Л	И	Л	Л
справа стена	Л	И	Л	Л	Л	И	Л	Л
снизу свободно	Л	Л	И	Л	Л	И	И	Л
слева стена	Л	Л	Л	И	Л	Л	И	И
клетка чистая	И	Л	Л	Л	И	Л	Л	И

104 Вот буквы армянского алфавита. Найди три одинаковые буквы, напиши такую букву в тетради.

ե պ հ տ կ ղ ց ի վ ա բ զ ը մ
ո ս բ զ մ ո ն ղ խ ն լ ղ տ կ
ի ա լ ղ ի խ ը ս հ վ ց ե պ

105 Нарисуй состояние *Робота* после выполнения им алгоритма `переход` из данного начального состояния.



```
алг переход
  дано |
  надо |
нач
  влево
  влево
  если снизу
       свободно
    то вниз
       вправо
       вправо
  все
  вправо
  закрась
  вправо
  закрась
  вниз
  если снизу
       свободно
    то вниз
    иначе влево
          вниз
  все
кон
```

106 Построй такую последовательность партии игры *Крестики-нолики*, в которой на седьмом ходу выигрывает Первый.

107 Реши задачу.
Разложи по семи кошелькам 127 рублёвых монет так, чтобы любую сумму от 1 до 127 р. можно было выдать, не открывая кошельков (другими словами, отдав всё содержимое одного или нескольких кошельков).

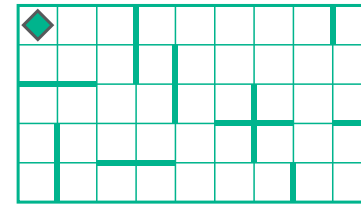
108 Построй состояние *Робота* после выполнения каждого алгоритма из каждого из данных начальных состояний. Построй

таблицу по образцу — в каждую клетку наклей заготовку поля, раскрась клетки, нарисуй положение *Робота.*

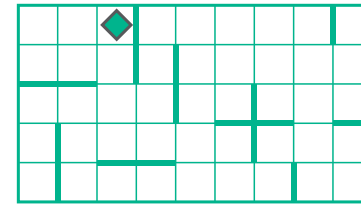
```
алг А
  дано |
  надо |
нач
  нц 6 раз
    если справа свободно
      то вправо
    все
    если снизу свободно
      то вниз
    все
  кц
кон
```

```
алг Б
  дано |
  надо |
нач
  нц 6 раз
    если снизу свободно
      то вниз
    все
    если справа свободно
      то вправо
    все
  кц
кон
```

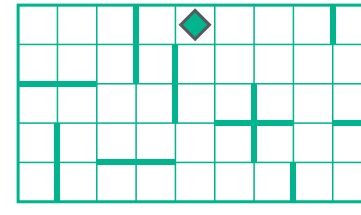
Начальное состояние 1:



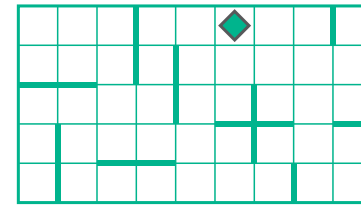
Начальное состояние 2:



Начальное состояние 3:



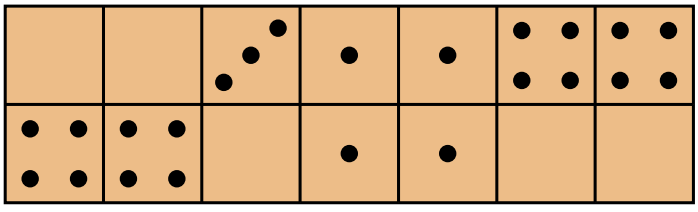
Начальное состояние 4:



Начальное состояние	Состояние после выполнения алгоритма А	Состояние после выполнения алгоритма Б

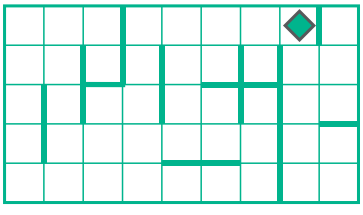
109 Несколько костяшек из одного набора домино уложили так, как показано на рисунке. Определи, где проходят границы между костяшками. Для этого вырежи такой же рисунок со вкладыша тетради проектов, наклей его в тетрадь и проведи на нём эти границы жирными линиями.

Нужно иметь в виду, что в одном наборе домино все костяшки разные.

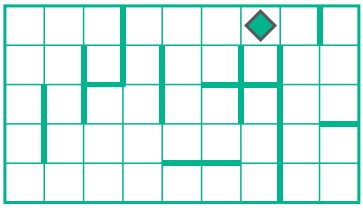


110 Составь такой алгоритм для *Робота,* после выполнения которого из каждого из данных начальных состояний *Робот* окажется в левом нижнем углу поля.

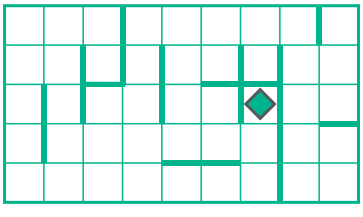
Начальное состояние 1:



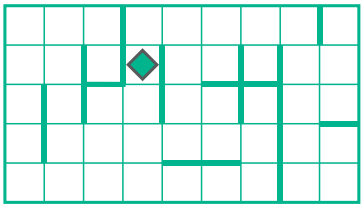
Начальное состояние 2:



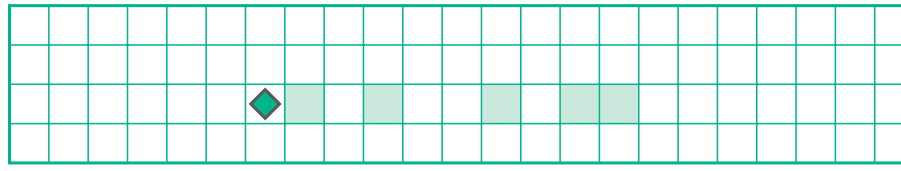
Начальное состояние 3:



Начальное состояние 4:



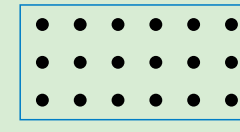
111 На поле *Робота* стен нет. В ряду правее той клетки, где стоит *Робот*, некоторые клетки закрашены, причём самая дальняя закрашенная клетка находится не дальше чем в 10 шагах от *Робота*, например, как на рисунке ниже. Какие именно клетки закрашены изначально, неизвестно, известно только, что больше закрашенных клеток на поле нет.



Составь алгоритмы, которые заставляют *Робота*:
а) закрасить клетку на шаг ниже каждой из закрашенных изначально;
б) закрасить клетку на шаг выше и на шаг ниже каждой из закрашенных изначально;
в) закрасить клетку на шаг левее каждой из закрашенных изначально;
г) закрасить клетку на шаг правее каждой из закрашенных изначально.

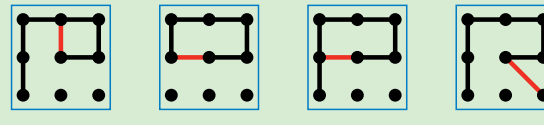
112 Даны правила игры *Ползунок*.

Правила игры *Ползунок*



Начальная позиция. Игровое поле состоит из нескольких рядов точек, расположенных в прямоугольнике. Пример такого поля приведён справа.

Возможные ходы. На первом ходу Первый соединяет горизонтальным или вертикальным отрезком две любые соседние точки. На каждом из следующих ходов игрок проводит горизонтальный или вертикальный отрезок, соединяющий один из концов получившейся до этого ломаной линии с какой-нибудь соседней точкой, через которую линия ещё не прошла. Например, такие ходы не разрешены:



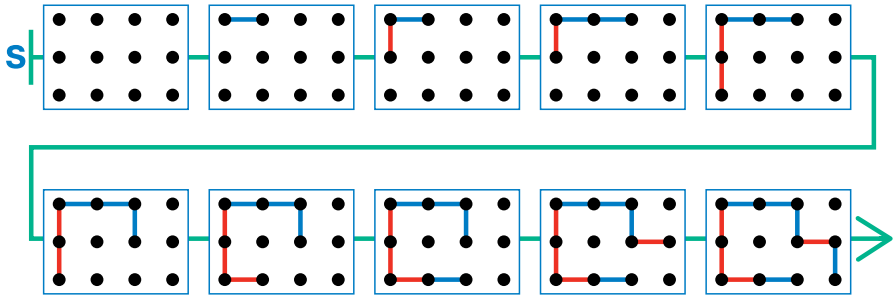
Как определить победителя. Игра заканчивается, если очередной ход сделать нельзя. Выигрывает игрок, который сделал последний ход.

Рассмотри последовательность S позиций партии игры *Ползунок* на поле 4×3 точки. Ходы Первого помечены синим, ходы Второго — оранжевым. Ответь на вопросы:

а) Сколько ходов было сделано в этой партии?

б) Сколько ходов сделал Первый и сколько — Второй?

в) Кто выиграл?

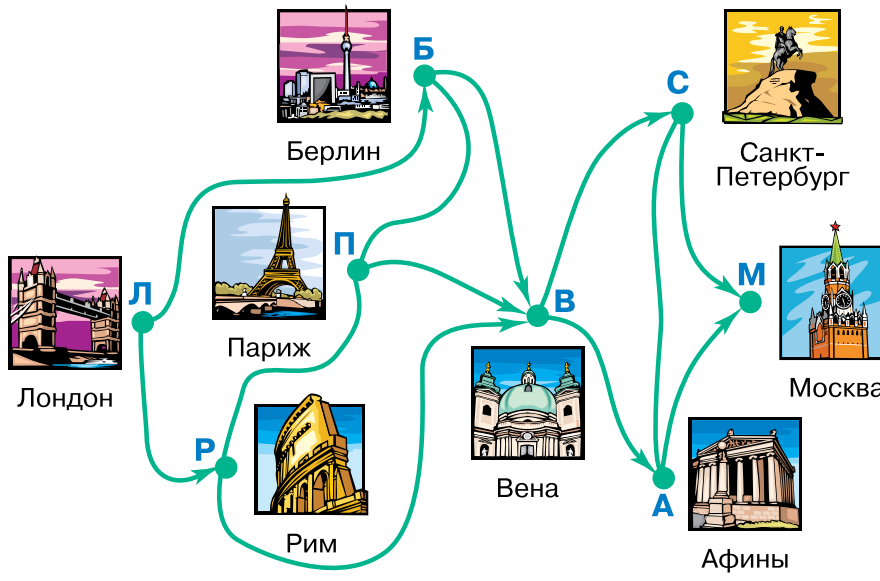


113 Придумай такую партию игры *Ползунок* на поле 4×3, в которой выиграл Второй. Построй последовательность позиций этой партии. Заготовки полей есть на вкладыше тетради проектов. Теперь построй последовательность P позиций такой партии, в которой выиграл Первый.

114 Построй дерево, для которого все утверждения в таблице имеют указанные истинностные значения.

Имя	Утверждение	Значение
A	Все листья этого дерева — элементы третьего уровня.	И
B	Все бусины в этом дереве только двух цветов — чёрные и белые.	И
C	В этом дереве есть две одинаковые последовательности.	Л
D	Каждый элемент этого дерева — круглая бусина.	И
E	В этом дереве меньше восьми последовательностей.	Л

115 Определи по графу, сколькими способами можно проехать из города Л в город М. По дорогам со стрелками можно двигаться только в направлении стрелок, по каждой дороге можно проезжать не больше одного раза.



Выигрышная стратегия

Мы строили различные партии игр, но при этом совсем не принимали во внимание стремление игроков к победе. Теперь нас будут интересовать лишь такие партии, в которых *оба игрока стремятся к победе*. Будем называть такие партии *разумными*. Игроков, которые стараются победить, а не делают ходы наугад, мы тоже будем называть *разумными*.

Итак, играют двое и каждый из них стремится к победе. Если правила игры не допускают ничьей, то в каждой партии кто-то из игроков обязательно выигрывает. Оказывается, что в каждой игре с полной информацией, правила которой не допускают ничьей, существует *выигрышная стратегия* для одного из игроков.

Выигрышная стратегия — это правило, следуя которому один из игроков может выиграть, как бы ни играл его противник.