МАТЕРИАЛЫ ЗАДАНИЙ

олимпиады школьников «ЛОМОНОСОВ» по информатике

2014/2015 учебный год

http://olymp.msu.ru

Задания отборочного этапа олимпиады школьников «Ломоносов-2015» по информатике (5–9 классы)

Задача 1. Системы счисления

Сколько единиц содержится в двоичном представлении шестнадцатеричного числа 3de644638d7? Ответ запишите одним числом.

Задача 2. Шахматы

На шахматной доске 8x8 в позиции (a, b) стоит белый слон, в позиции (c, d) стоит черный король. Пусть a, b, c и d – натуральные числа от 1 до 8 – номера вертикалей и горизонталей, соответственно.

Запишите логическое условие (либо на каком-либо языке программирования, либо на естественном языке), которое является истинным в том случае, если какая-либо фигура бьет другую, и ложным в противном случае. В записи условия разрешается использовать только арифметические операции, операции сравнения, логические связки.

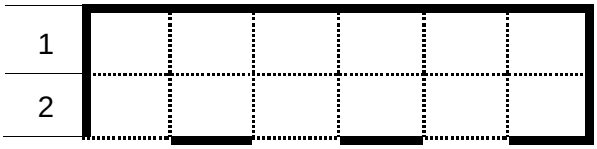
Задача 3. Робот

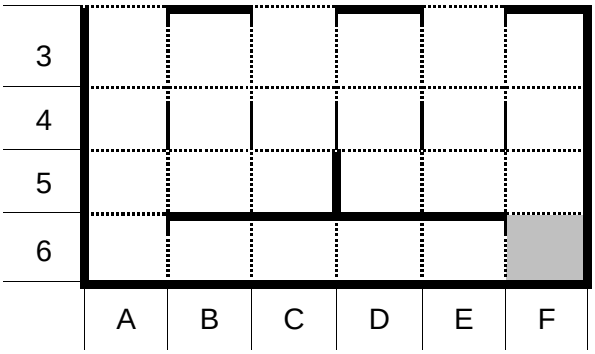
Робот ТРОТРО оказался в лабиринте. Лабиринт представляет собой клетчатое прямоугольное поле со стенками. В лабиринте одна из угловых клеток является выходом из лабиринта.

ТРОТРО может видеть только ближайшие 8 клеток вокруг него, если они не закрыты стенками. Связь с ТРОТРО потеряна, поэтому он действует сам на основе программы, зашитой в него на этот случай разработчиками.

Нужно посчитать количество клеток лабиринта, включая клетку-выход, таких, что, начав движение в ней, ТРОТРО, двигаясь по своей программе, уцелеет и выйдет из лабиринта. Если ТРОТРО начнёт движение в сторону находящейся рядом с ним стены, то он разрушится и программа прервётся.

Ниже приведен лабиринт и программа ТРОТРО. Верх лабиринта - горизонталь 1, низ лабиринта - горизонталь 6, левая сторона лабиринта - вертикаль A, правая сторона лабиринта - вертикаль F. Выход из лабиринта помечен закрашенной клеткой (клетка F6).





Программа:

```
НАЧАЛО
  ПОКА снизу свободно ИЛИ справа свободно
     ПОКА снизу свободно
         вниз
     КОНЕЦ ПОКА
     ЕСЛИ справа свободно
     ТО вправо
     КОНЕЦ ЕСЛИ
  КОНЕЦ ПОКА
КОНЕЦ
```

Задача 4. Шифр

К вам попал зашифрованный текст, означающий большую истину для многих программистов. Расшифруйте его, т.е. предложите способ восстановления исходного текста и найдите этот текст. В ответе запишите сначала расшифрованный текст, а затем алгоритм расшифровки известным вам способом.

```
vujgvmCfb tj ufscfu ouib z/vhm
jdjuFyqm jt fscfuu uibo jdju/jnqm
fTjnqm tj scfuuf ibou fy/dpnqm
yDpnqmf jt cfuufs boui dbufe/dpnqmj
uGmb tj fuufsc ouib oftufe/
bstfTq jt uufscf uibo otf/ef
uzSfbebcjmj vout/dp
djbmTqf dbtft (ubsfo djbmtqf hifopv up csfbl ifu t/svmf
ipvhiBmu zqsbdujdbmju fbutc uz/qvsj
Fsspst tipvme wfsof qbtt foumz/tjm
```

```
omfttV mjdjumzfyq odfe/tjmf
Jo fui dfgb pg hvjuz-bncj gvtfsf fui ubujpoufnq up ftt/hv
Uifsf vmetip fc pof.. boe sbcmzqsfgf zpom pof pvt..pcwj xbz pu pe
ju/
Bmuipvhi uibu bzx bzn puo cf wjpvtpc bu jstug ttvomf sfzpv( i/Evud
xOp tj scfuuf ibou /ofwfs
uipvhiBm fsofw jt fopgu cfuufs boui iu++sjh x/op
gJ ifu nfoubujpojnqmf tj eibs pu mbjo-fyq tju( b bec /jefb
Jg fui foubujpojnqmfn jt fbtz up bjo-fyqm ju znb cf b hppe jefb/
bnftqbdftO bsf pof ipoljoh sfbuh efbj .. fu(tm pe psfn gp tf"uip
```

Задача 5. Малыш и Карлсон

Малыш и Карлсон устали гоняться за привидениями по крышам ночного Стокгольма и сели поиграть в игру с числами. Карлсон загадал 4-значное число с неповторяющимися цифрами, а Малыш сделал четыре попытки разгадать это число. Каждый раз Малыш называет некоторое 4-значное число с неповторяющимися цифрами, а Карлсон сообщает в ответ два числа, M и N. M означает количество цифр, которые угадал Малыш, но они находятся на другой позиции в числе Карлсона. N означает количество цифр, которые угадал Малыш на тех же позициях, на которых они находятся в числе Карлсона.

Например: Карлсон загадывает число «3219». Малыш отвечает: «2310». Карлсон ему в ответ: M = 2 («2» и «3» — угаданы на неверных позициях) и N = 1 (одна цифра «1» угадана вплоть до позиции).

Малыш сделал следующие попытки:

```
1723: M = 0, N = 1

5692: M = 2, N = 0

7250: M = 1, N = 0

6789: M = 2, N = 0

4213: M = 0, N = 1
```

Сколько различных чисел мог загадать Карлсон, чтобы они удовлетворяли этим попыткам Малыша?

Задача 6. Бармаглот-1

Компания *Barmaley's Computing* выпустила новый компьютер *Barmaglot*. Поскольку у сотрудников компании несколько странные представления о технологиях, компьютер оказался довольно непривычным.

Во-первых, исходные данные он читает с ленты, где могут быть записаны целые числа и латинские буквы. Чтобы показать, что ввод данных закончен, ленту нужно просто оторвать.

Во-вторых, результаты вычислений компьютер печатает на точно такой же ленте. Центральный процессор компьютера оснащён одним регистром; вместо оперативной памяти компьютер оснащён

очередью и стеком неограниченного размера. Регистр, а также каждая ячейка очереди и стека, могут содержать либо число, либо латинскую букву или пробел, либо специальное значение `[BARMALEY]`, которое используется для обозначения логической лжи, любое другое значение обозначает истину.

Программа для компьютера *Barmaglot* представляет собой строку символов, каждый из которых задаёт машинную команду; программа, состоящая из символов-команд, выполняется последовательно слева направо, кроме двух команд, которые могут нарушить эту последовательность.

- Команды `a`, `b`, `c` и все остальные латинские буквы означают "занести данную букву в регистр";
- Команды `0`, `1`, ..., `9` означают "занести в регистр соответствующее число";
- Команда `@` означает занести в регистр пробел;
- Команды `+`, `-`, `*`, `/` означают соответствующие целочисленные арифметические действия;
- Команды `<`, `>`, `=` означают сравнение двух чисел или двух символов;
- Команды `&` и `|` означают логическое "и" и логическое "или";

Все указанные выше арифметические команды используют значение из регистра в качестве левого операнда, значение с вершины стека в качестве правого (оно при этом из стека извлекается), результат заносится обратно в регистр.

- Команда `#` умножает содержимое регистра в десять раз;
- Команда `_` делит содержимое регистра в десять раз;
- Команда `!` работает как логическое отрицание содержимого регистра: если там значение `[BARMALEY]`, то оно заменяется на 1, любое другое заменяется на `[BARMALEY]`;
- Команда `.` выдаёт текущее значение регистра на печать;
- Команда `?` вводит очередное число или букву в регистр, а если на вводе кончилась (оборвалась) лента, заносит в регистр значение `[BARMALEY]`;
- Команда `]` заносит значение из регистра в стек;
- Команда `[` извлекает значение с вершины стека и заносит его в регистр; если извлекать нечего, в регистр заносится `[BARMALEY]`;
- Команда `~` меняет местами значения в регистре и на вершине стека.
- Команда `}` заносит значение из регистра в очередь,
- Команда `{` извлекает из очереди самое старое значение и помещает в регистр, если очередь не пуста; и помещает в регистр `[BARMALEY]`, если очередь пуста.
- Команды `(` и `)` предназначены для организации ветвлений и циклов и всегда должны в программе стоять парами, то есть в программе должен обязательно соблюдаться баланс круглых скобок. Выполняются они так. Если в регистре находится `[BARMALEY]`, то команда `(` идёт по программе вперёд, пока не найдёт парную команду `)`, и после этого выполнение продолжится со следующей за этой

закрывающей скобкой позиции; если в регистре было что-то другое, команда `(` ничего не делает, то есть выполнение продолжается прямо с команды, следующей за ней. Команда `)`, наоборот, если в регистре `[BARMALEY]`, не делает ничего, тогда как если там что-то другое, просматривает программу назад до парной скобки `(`, после чего продолжает выполнение с команды, стоящей после такой скобки справа.

- Команда `"` прекращает выполнение программы, при этом выполнение считается успешным. Если программа кончилась, не встретив эту команду, она завершается аварийно.

Пробелы в программе игнорируются.

Пример программы, которая печатает традиционную строчку “HELLO WORLD”:

```
H.E.L.L.O.@.W.O.R.L.D."
```

В вашем распоряжении оказался эмулятор данного компьютера и требуется написать для него программу, вводящую натуральное число 0 < N < 10 с ленты и печатающую соответствующее число букв Z на ленту. Например, для ввода 5 на ленте должна быть выведена последовательность ZZZZZ.

Задача 7. Бармаглот-2

В вашем распоряжении оказался эмулятор компьютера Barmaglot (см. предыдущую задачу). Требуется написать для него программу, вводящую натуральное число N с ленты и печатающую последовательность чисел от 1 до N. Например, для ввода 9 на ленте должна быть выведена последовательность `1 2 3 4 5 6 7 8 9`. Числа выводятся через пробел.

Задача 8. Системы счисления-2

Сколько единиц содержится в двоичном представлении шестнадцатеричного числа, содержащегося в заданном файле? (размер файла – 20МБ) Ответ запишите одним числом.

Задания отборочного этапа олимпиады школьников «Ломоносов-2015» по информатике (10–11 классы) Первый тур

Задачи 1, 2, 3, 5, 6, 7 имеют вариативную постановку. Каждому участнику предлагался один из вариантов этих задач. Комментарии по этим вариантам приводятся ниже. Для приема решений использовалась система Ejudge.

Задача 1. Упаковка числа

Рассмотрим способ упакованной записи чисел в двоичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 1111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в двоичной системе и закрывающей скобки –]. Так число 111111100 в упакованной записи будет записано следующим образом: 1[1[1[10]]]0[10]

Найдите наименьшую по длине упакованную запись двоичного числа
11100000000100000111111111110

Задача будет оценена в зависимости от верности упаковки и длины получившейся последовательности.

Примечание: в других вариантах этого задания предлагались разные длинные числа и основания систем счисления (двоичная или троичная).

Задача 2. Последовательность

Последовательность чисел 38, 37, 36, 35, 39, 34, 33, 32, 31, 30, ... , 100 была получена выписыванием 500 первых натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в обратном алфавитном порядке. Определите число, стоящее в последовательности на 114-м месте.

Указание: При решении задачи считайте, что в записи римскими цифрами цифра I может предшествовать только цифрам V и X, цифра V может предшествовать только цифре X, цифра X может предшествовать только цифрам L и C, цифра L может предшествовать только цифре C, цифра C может предшествовать только цифрам D и M. Для решения можно использовать электронные таблицы.

При сдаче решения в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.).

Примечание: в других вариантах предлагались другие последовательности.

Задача 3. Битовые карты

Квадратная битовая карта A представляет собой матрицу размера n×n, любой элемент a[i, j] которой может принимать значение либо 0, либо 1. В задаче будем использовать следующие простейшие операции над битовыми картами:

- INV(A) – результатом является битовая карта B, в которой b[i, j] = 1, если a[i, j] = 0, и b[i, j] = 0, если a[i, j] = 1;
- MAX(A, B) – результатом является битовая карта C, в которой c[i, j] = max(a[i, j], b[i, j]);
- MIN(A, B) – результатом является битовая карта C, в которой c[i, j] = min(a[i, j], b[i, j]);
- TRANS(A) – результатом является битовая карта B, в которой b[i, j] = a[j, i];
- MASK(A) – результатом является битовая карта B, в которой b[i, j] = 0, если $i \le n/2$, $j \ge n/2$, и b[i, j] = a[i, j] при всех остальных i, j.

Из простейших операций можно составлять составные операции. Например, составная операция MIN(A, INV(TRANS(A))) применённая к следующей карте A:

```
1 0

1 1
```

даст в результате карту:

```
0 0

1 0
```

Запишите составную операцию, применение которой к произвольной неизвестной битовой карте X размера 6x6 даст в результате битовую карту:

```
0 0 0 1 1 1

0 0 0 1 1 1

0 0 0 1 1 1

1 1 1 0 0 0

1 1 1 0 0 0

1 1 1 0 0 0
```

Запись искомой составной операции должна быть как можно меньшей длины.

Примечание: в других вариантах изменялся список простейших операций и итоговая битовая карта.

Задача 4. Упаковка числа - 2

Рассмотрим способ упакованной записи чисел в p-ичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 111111111, заменять на следующую конструкцию. Сначала идет повторяемая цифра (в примере – 1), затем открывающаяся скобка перед числом повторений [, затем упакованная записи числа повторений в p-ичной системе и, наконец, закрывающаяся скобка]. Так число 111111111000 в упакованной записи в двоичной системе может быть записано следующим образом: 1[10[10]1]0[11].

Составьте программу, которая упаковывает число в p-ичной системе счисления. Длина упакованного числа должна быть минимальной.

В первой строке вводится основание системы счисления p (1 < p < 11). Во второй строке находится число, записанное в p-ичной системе счисления, длина записи не превышает 1000 символов.

Ввод	Вывод
2	1[1010]000

Пример:	1111111111000	

Задача 5. Пьюникод

Восстановите связное сообщение по тексту, который был получен после некоторого преобразования исходного текста:

xn--punycode----------ascii--ujtaacbaerdf0bebiat2bbie8aged5ehg8cep7jivfj1bdsvfbbifadk3agcebcdgldfkb1by2bcwd8ikh1a8cf2y4fmpub4he2w1a6t

Примечание: в других вариантах предлагался другой шифрованный текст.

Задача 6. Уравняй числа

Результатом применения операции 1 к паре натуральных чисел (a, b) является пара натуральных чисел (a+1, 2×b). Результатом применения операции 2 к паре натуральных чисел (a, b) является пара натуральных чисел (2×a, b+1). Рассмотрим пару (3, 7). Применяя к паре данные операции, можно получить пару равных чисел: (3, 7) —^{1}→ (4, 14) —^{2}→ (8, 15) —^{2}→ (16, 16).

Требуется за как можно меньшее число шагов при помощи операций 1 и 2 получить пару равных чисел из пары (63, 14). Результат запишите в виде последовательности из единиц и двоек (номеров операций) без пробелов. Если "уравнять" данную пару нельзя, укажите результат 0. Начиная со второй строки ответа, опишите, как было получено решение.

Примечание: в других вариантах предлагалась другая начальная пара чисел.

Задача 7. КНБ

Альберт, Бертольд и Корвин играют втроём в игру "камень-ножницы-бумага". По счёту "три" все трое одновременно показывают рукой один из трёх возможных жестов, означающих камень (Rock), ножницы (Scissors) или бумагу (Paper). Считается, что камень побеждает ("затупляет") ножницы, ножницы побеждают ("разрезают") бумагу, а бумага побеждает ("заворачивает") камень.

На каждом кону разыгрывается одно очко. Если все трое показывают один и тот же жест, считается, что произошла ничья, и все получают по 1/3 очка. Если все трое показывают разные жесты, "победив" друг друга по кругу, тоже фиксируется ничья с таким же результатом. Если двое показывают одинаковый жест, и третий побеждает их обоих, то он получает одно очко, а остальные – ничего. Если двое, показав одинаковый жест, побеждают третьего, то они получают по 1/2 очка, а проигравший – ноль.

Протоколом игры называется последовательность трёхбуквенных слов, разделённых пробелами и/или переводами строки. Каждое слово протокола описывает исход одного кона, причём первая буква соответствует жесту Альберта, вторая – жесту Бертольда, третья – жесту Корвина. Камень кодируется буквой R, ножницы – буквой S, бумага – буквой P. Например, слово SRR означает, что

Альберт показал "ножницы", а Бертольд и Корвин показали "камень" и, таким образом, получили по 1/2 очка.

Скачайте zip-архив с протоколом (предлагалась ссылка) и подсчитайте, сколько очков получил каждый из игроков в итоге всей игры, отражённой в протоколе. Ответ выдайте в виде трёх строк. В первой строке должен быть результат Альберта, во второй – результат Бертольда, в третьей – результат Корвина. Каждый результат запишите в виде: целая часть числа, пробел, а затем простая несократимая дробь в виде числителя и знаменателя, разделённых дробной чертой. Если дробная часть равна нулю, то не указывайте её в ответе. Если целая часть равна нулю, а дробная не равна, то не указывайте нулевую целую часть в ответе.

Пример протокола:

```
SRR PPP RSP PPS PSP RPP RPP SSR
```

Пример ответа:

```
2/3

3 1/6

4 1/6
```

Примечание: в других вариантах предлагались другие протоколы.

Задача 8. Бармаглот-1

В вашем распоряжении оказался эмулятор компьютера Barmaglot (см. задачу 6 для 5-9 классов). Требуется написать для него программу, которая по заданному на ленте году выводит строку YES, если он високосный, и NO в противном случае. Год является високосным в двух случаях: либо он кратен 4, но при этом не кратен 100, либо кратен 400. Год не является високосным, если он не кратен 4, либо он кратен 100, но при этом не кратен 400.

Задача 9. Бармаглот-2

В вашем распоряжении оказался эмулятор компьютера Barmaglot (см. задачу 6 для 5-9 классов). Требуется написать для него программу, которая находит и печатает максимальное число, среди записанных на ленте. Числа на ленте неотрицательные и разделены ровно одним пробелом.

Задача 10. Дороги

Иван Васильевич является владельцем компании «В добрый путь», которая занимается ремонтом старых дорог и строительством новых. Правда, в последнее время дела идут плохо - никто не обращается к ним с просьбой что-то починить или построить.

Казалось бы, банкротства не избежать, но на днях появилось предложение, от которого Иван Васильевич не может отказаться. А именно, руководство города Байттаун объявило конкурс на реконструкцию дорог! Замечательная возможность поправить свое положение!

Всего в городе M = 1000 дорог. Предприятие «Дураки и дороги» умеет делать N = 1000 различных операций над каждой дорогой.

Вдохновленный этим предложением, Иван Васильевич вместе с главным инженером придумали K = 50 планов реконструкции дорог. Архив с планами доступен по ссылке (предлагалась ссылка). Каждый план представляет собой матрицу размера MxN, состоящей из 0 и 1, причем на пересечении i-й строки и j-го столбца стоит 1, если, согласно плану, нужно выполнить j-ю операцию над i-й дорогой, и 0 в противном случае.

Осталось только выбрать план, с которым нужно идти к заказчику. Но вот незадача - конкуренты из компании «Дорожная карта» умудрились украсть один из K вариантов и теперь собираются с ним идти на конкурс. Но программисты «Дураков и дорог» тоже не лыком шиты: они смогли получить ограниченный доступ к файлу на сервере конкурентов.

В связи с этим Иван Васильевич обращается к вам с просьбой помочь определить, какой именно вариант был украден, чтобы не идти с ним к заказчику.

Правда, вы не можете просто так посмотреть файл на сервере, иначе программисты «Дорожной карты» обнаружат подозрительную активность и перезагрузят сервер. Вместо этого вы можете узнать следующую информацию: в скольких ячейках какой-то подматрицы размера 5x5 лежит 1.

Решение должно выводить запросы на стандартный поток ввода. Каждый запрос выводится в отдельной строке. Существует два типа запросов:

- '? i j' - запрос информации о содержимом файла число единиц в подматрицы размера 5x5 с левым верхнем углом в ячейке i j. В ответ на этот запрос на стандартный поток ввода поступит одно число - результат запроса. Запрашиваемая подматрица не должна выходить за границы исходной матрицы. Элементы нумеруются с нуля.
- '! ans' - программа угадала номер файла (ans). После этого запроса решение должно завершиться.

На каждый запрос '?' вводится число единиц в соответствующей подматрице.

После вывода каждого запроса решение должно выталкивать данные из буфера вывода с помощью следующих команд:

- На языке C: `fflush(stdout);`
- На языке Pascal: `flush(output);`
- На языке Python: `sys.stdout.flush();`
- На языке C++: `cout.flush();`
- На языке Java: `System.out.flush().`

Ниже приводится пример взаимодействия решения с сервером.

Запросы	**Ответы**
`? 0 0`	`10`

? 1 1 ! 1	8

Задания отборочного этапа олимпиады школьников «Ломоносов-2015» по информатике (10–11 классы) Второй тур

Задачи 1, 2, 5, 7 имеют вариативную постановку. Каждому участнику предлагался один из вариантов этих задач. Комментарии по этим вариантам приводятся ниже. Для приема решений использовалась система Ejudge.

Задача 1. Сообщение

Восстановите связное сообщение по тексту, который был получен после некоторого преобразования исходного текста:


```
&amp;#x0041;&amp;#x0020;&amp;#x006e;&amp;#x0075;&amp;#x006d;&amp;#
x0065;&amp;#x0072;&amp;#x0069;&amp;#x0063;
&amp;#x0020;&amp;#x0063;&amp;#x0068;&amp;#x0061;&amp;#x0072;&amp;#
x0061;&amp;#x0063;&amp;#x0074;&amp;#x0065;
&amp;#x0072;&amp;#x0020;&amp;#x0072;&amp;#x0065;&amp;#x0066;&amp;#
x0065;&amp;#x0072;&amp;#x0065;&amp;#x006e;
&amp;#x0063;&amp;#x0065;&amp;#x0020;&amp;#x0069;&amp;#x006e;&amp;#
x0020;&amp;#x0048;&amp;#x0054;&amp;#x004d;
&amp;#x004c;&amp;#x0020;&amp;#x0072;&amp;#x0065;&amp;#x0066;&amp;#
x0065;&amp;#x0072;&amp;#x0073;&amp;#x0020;
&amp;#x0074;&amp;#x006f;&amp;#x0020;&amp;#x0061;&amp;#x0020;&amp;#
x0063;&amp;#x0068;&amp;#x0061;&amp;#x0072;
&amp;#x0061;&amp;#x0063;&amp;#x0074;&amp;#x0065;&amp;#x0072;&amp;#
x0020;&amp;#x0062;&amp;#x0079;&amp;#x0020;
&amp;#x0069;&amp;#x0074;&amp;#x0073;&amp;#x0020;&amp;#x0055;&amp;#
x006e;&amp;#x0069;&amp;#x0076;&amp;#x0065;
&amp;#x0072;&amp;#x0073;&amp;#x0061;&amp;#x006c;&amp;#x0020;&amp;#
x0043;&amp;#x0068;&amp;#x0061;&amp;#x0072;
&amp;#x0061;&amp;#x0063;&amp;#x0074;&amp;#x0065;&amp;#x0072;&amp;#
x0020;&amp;#x0053;&amp;#x0065;&amp;#x0074;
&amp;#x002f;&amp;#x0055;&amp;#x006e;&amp;#x0069;&amp;#x0063;&amp;#
x006f;&amp;#x0064;&amp;#x0065;&amp;#x0020;
&amp;#x0063;&amp;#x006f;&amp;#x0064;&amp;#x0065;&amp;#x0020;&amp;#
```

x0070;oin
t.

Примечание: в других вариантах предлагались другие тексты.

Задача 2. Последовательность

Последовательность чисел 2303, 511, 3311, 1519, 2527, 735, 1743, 2751, 959, 1967, ... , 1792 была получена следующим образом. Сначала были выписаны 500 первых натуральных чисел, кратных 7, по возрастанию (7, 14, … 3500). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (007, 00E, …, DAC). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (700, E00, …, CAD). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FF8, FF1, …, 007). Затем строки были обратно преобразованы в исходные числа (2303, 511, …, 1792). Определите номер места, на котором стоит в последовательности число 1414 (места нумеруются с 1).

Указание: Для решения можно использовать электронные таблицы.

При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.).

Примечание: в других вариантах предлагались другие последовательности чисел.

Задача 3. Ёлочка

Перед Новым годом в Интернете появился сервис «Ёлочка». С его помощью можно завести себе web-страничку, на которой будет жить виртуальное дерево. Сначала высота ёлочки – 100 сантиметров. Каждые последующие 20 сантиметров зарабатываются просмотром странички в соответствии с последовательностью Фибоначчи. Т.е. для роста с 1 метра до 120 сантиметров нужен 1 просмотр, с 120 до 140 - 2 просмотра, с 140 до 160 потребуется 3 просмотра, с 160 до 180 уже 5 просмотров, а с 180 до 200 целых 8 просмотров. Таким образом, в результате 1 + 2 + 3 + 5 + 8 = 19 просмотров ваша ёлочка выросла до 2 метров. Ёлочка дискретная, поэтому её рост всегда кратен 20 см.

Вы поместили ссылку на ёлку в своей социальной сети. Некоторые ваши друзья посетили страничку, а некоторые поместили ссылку у себя (кстати, размещение ссылки на другой странице приравнивается к 5 просмотрам). 31 декабря рано утром (7:00:00) страничка была заблокирована (появился пароль), а к вам на почту пришел журнал посещений вашего дерева. Нужно вычислить, какой высоты стала ёлочка, и тогда страница вновь станет доступна (паролем является высота ёлочки в сантиметрах).

Модель роста ёлки, описанная в задаче, может не совпадать с моделью реального мира.

Формат журнала посещения:

1) время посещения страницы (2014-12-28 12:51:00)
2) тип действия: просмотр W, размещение на другой страничке S

Пример ввода:

```
2014-12-28 16:15:14 W

2014-12-28 18:37:01 S

2014-12-29 12:43:42 W

2014-12-30 14:05:18 W

2014-12-30 14:07:12 S

2014-12-30 19:02:36 W
```

Ответ к примеру:

```
180
```

Требуется написать программу, которая по логу определяет рост ёлочки в сантиметрах. Ввод-вывод осуществляется при помощи стандартных потоков или из/в файлы input.txt/output.txt.

Задача 4. Сравнение чисел

Рассмотрим способ упакованной записи чисел в двоичной системе. Идея способа состоит в том, чтобы некоторые одинаковые подряд идущие цифры заменять на конструкцию, состоящую из повторяемой цифры, открывающей скобки перед числом повторений - [, упакованной записи числа повторений в двоичной системе и закрывающей скобки -].

Так все следующие записи соответствуют упакованной версии числа 111111100:

```
111111100

1[10]1[10]1[10]100

11111110[10]

1[111]00

1[111]0[10]

1[1[11]]00

1[1[11]]0[10]

1[1[1[10]]]00

1[1[1[10]]]0[10]
```

Требуется написать программу, сравнивающую два числа в упакованном виде. Числа даны на стандартном потоке ввода каждый в своей строке без ведущих нулей. Гарантируется, что изначальные (распакованные) числа больше нуля и не превосходят 2^100000.

В стандартный поток вывода необходимо напечатать <, если первое число меньше второго; =, если числа равны и >, если первое число больше второго.

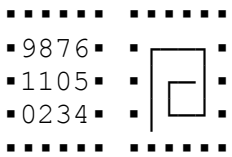
Примеры:

Ввод	Вывод
111111100 1[1[1[10]]]0[10]	=
11 10	>
1[10] 101	<

Задача 5. Найди питона

Представим, что последовательность из N десятичных цифр описывает питона длины N. Для описания питона будем использовать первые N цифр последовательности 012345678910111213141516171819202..., образованной выписыванием подряд цифр из записи неотрицательных натуральных чисел в порядке возрастания. 0 - первая цифра последовательности - соответствует голове питона, N-ная - хвосту.

На плоскости питон всегда сворачивается в спираль специальным образом. Голова свернувшегося питона всегда находится у центра спирали. Туловище свернувшегося питона следует либо по ходу часовой стрелки, если рассматривать его от головы к хвосту, либо против хода часовой стрелки. На рисунке показан свернувшийся питон 012345678910 длины 12. Рядом изображена схема, как свернулся этот питон.



Все питоны всегда сворачиваются по такой схеме плотно, без пропусков. Голова питона на рисунке справа направлена вправо. Также допускается, что питон сворачивается так, что его голова направлена вверх, влево или вниз.

Ваша задача: на заданным матрицам M, состоящей из K строк длиной K десятичных цифр (K > N), найти свернувшегося в спираль питона P, имеющего самую большую длину среди всех питонов, находящихся на матрице, и выдать в результате координаты головы питона (номер строки, номер столбца) и длину питона. Учтите, что питоны в матрице могут пересекаться.

Пример:

```
K: 7

M:
```

```
1111210

1023430

1110510

1987640

1812700

0703000

0654000
```

Результат:

```
(3, 4)

20
```

Вам дается три файла с питонами. Требуется получить ответ для каждого из них по две строки, соответственно (всего 6 строк). После 6-й строки опишите, как был получен ответ (это может быть текст программы или словесное описание).

Примечание: в разных вариантах предлагались разные файлы с питонами.

Задача 6. Уравняй числа

Результатом применения операции 1 к паре натуральных чисел (a, b) является пара натуральных чисел (a+1, 2×b). Результатом применения операции 2 к паре натуральных чисел (a, b) является пара натуральных чисел (2×a, b+1). Рассмотрим пару (3, 7). Применяя к паре данные операции, можно получить пару равных чисел: (3, 7) —^{1}→ (4, 14) —^{2}→ (8, 15) —^{2}→ (16, 16).

Требуется за как можно меньшее число шагов с помощью операций 1 и 2 получить из пары (14, 20) пару равных чисел.

Результат запишите в виде последовательности из единиц и двоек (номеров операций) без пробелов. Если «уравнять» данную пару нельзя, укажите результат 0. Начиная со второй, строки опишите, как было получено решение.

Задача 7. Рыцари круглого стола

2n+1 рыцарей Круглого стола всегда сидят за Круглым столом таким образом, что для каждого рыцаря справедливо свойство: рыцарь побеждает в поединке любого из n рыцарей, сидящих от него через 1, 3, 5, 2n-1 стульев, если считать стулья по часовой стрелке, и проигрывает в поединке любому из n рыцарей сидящих на оставшихся стульях.

Рыцари провели турнир, состоящий из нескольких поединков, в каждом из которых сражались по два рыцаря. Результаты они занесли в манускрипт. В первой строке манускрипта указано количество рыцарей. Во второй строке манускрипта указаны имена рыцарей, разделённые пробелами, перечисленные по порядку рассадки за Круглым столом по часовой стрелке, имя любого рыцаря

состоит из не более 200 латинских букв. В третьей строке манускрипта указано M – количество поединков в турнире. Оставшиеся M строк манускрипта содержат по одной паре имён рыцарей, разделённых пробелами. Каждая пара имён указывает рыцарей, сошедшихся в поединке. Не бывает поединков, в которых рыцарь сражается сам с собой.

По заданному манускрипту (дается ссылка на файл) определите итоговый счёт, учитывая, что за победу в поединке рыцарь получает 2 очка, а за поражение 0 очков.

Записывая ответ, упорядочите имена рыцарей по невозрастанию набранных в турнире очков. Имена рыцарей, набравших одинаковое количество очков, упорядочите по алфавиту. Каждое имя расположите на отдельной строке. После имени через пробел в той же строке укажите набранные рыцарем очки.

Пример манускрипта:

```
5
lancelot galahad percival arthur lamorak
5
lancelot galahad
lancelot percival
lancelot arthur
lancelot lamorak
percival galahad
```

Пример ответа:

```
lancelot 4
arthur 2
galahad 2
percival 2
lamorak 0
```

Примечание: в других вариантах предлагались другие файлы с манускриптами.

Задача 8. Бармаглот-3

В вашем распоряжении оказался эмулятор компьютера Barmaglot (см. задачу 6 для 5-9 классов). Требуется написать для него программу, которая по заданной на ленте строке, состоящей из латинских символов, выводит её в перевернутом виде.

Задача 9. Бармаглот-4

В вашем распоряжении оказался эмулятор компьютера Barmaglot (см. предыдущую задачу). Требуется написать для него программу, которая для каждого числа заданной последовательности целых чисел (больших 1) выводит, является ли оно простым или нет (один символ Y, если число простое и N если нет).

Задача 10. Новые дороги

Иван Васильевич является владельцем компании «В добрый путь – 2», которая занимается ремонтом старых дорог и строительством новых. Правда, в последнее время дела идут плохо - никто не обращается к ним с просьбами что-то починить или построить.

Казалось бы, банкротства не избежать, но на днях появилось предложение, от которого Иван Васильевич не может отказаться. А именно, руководство города Байттаун объявило конкурс на реконструкцию дорог! Замечательная возможность поправить свое положение!

Всего в городе M = 1000 дорог. Предприятие «Дураки и дороги» умеет делать N = 1000 различных операций над каждой дорогой.

Вдохновленный этим предложением, Иван Васильевич вместе с главным инженером придумали K = 50 планов реконструкции дорог. Архив с планами доступен по ссылке (ссылка прилагается). Каждый план представляет собой матрицу размера MxN, состоящей из 0 и 1, причем на пересечении i-й строки и j-го столбца стоит 1, если, согласно плану, нужно выполнить j-ю операцию над i-й дорогой, и 0 в противном случае.

Осталось только выбрать план, с которым нужно идти к заказчику. Но вот незадача - конкуренты из компании «Дорожная карта» умудрились украсть один из K вариантов и теперь собираются с ним идти на конкурс. Но программисты «Дураков и дорог» тоже не лыком шиты - они смогли получить ограниченный доступ к файлу на сервере конкурентов.

В связи с этим Иван Васильевич обращается к вам с просьбой помочь определить, какой именно вариант был украден, чтобы не идти с ним к заказчику.

Правда, вы не можете просто так посмотреть файл на сервере, иначе программисты «Дорожной карты» обнаружат подозрительную активность и перезагрузят сервер. Вместо этого, вы можете узнать следующую информацию: чего больше в подматрице размера 5x5 - нулей или единиц.

Решение должно выводить запросы на стандартный поток ввода. Каждый запрос выводится в отдельной строке. Существует два типа запросов:

- '? i j' - запрос информации о содержимом файла число единиц в подматрицы размера 5x5 с левым верхнем углом в ячейке i j. В ответ на этот запрос на стандартный поток ввода поступит одно число - результат запроса. Запрашиваемая подматрица не должна выходить за границы исходной матрицы. Элементы нумеруются с нуля.
- '! ans' - программа угадала номер файла (ans). После этого запроса решение должно завершиться.

На каждый запрос '?' вводится "0", если нулей больше, и "1", если больше единиц в соответствующей подматрице.

После вывода каждого запроса решение должно выталкивать данные из буфера вывода с помощью следующих команд:

- На языке C: `fflush(stdout);`
- На языке Pascal: `flush(output);`
- На языке Python: `sys.stdout.flush();`
- На языке C++: `cout.flush();`
- На языке Java: `System.out.flush().`

Ниже приводится пример взаимодействия решения с сервером.

Запросы	**Ответы**

`? 0 0`
`? 1 1`
`! 1`
	`0`
	`1`

Задания заключительного этапа олимпиады школьников «Ломоносов-2015» по информатике (5–9 классы)

Задания выполнялись целиком в письменном виде.

Задача 1. Системы счисления

Переведите число 2211201122110201 из троичной системы счисления в систему счисления с основанием 27. В качестве цифр используйте десятичные цифры и заглавные латинские буквы.

Задача 2. Квадродерево

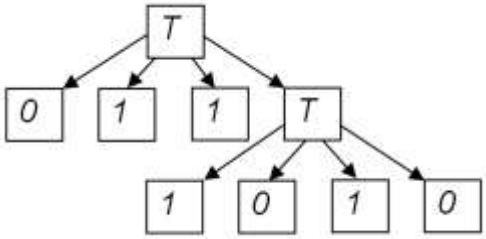
Рассмотрим способ представления растровых черно-белых квадратных изображений, называемый квадродеревом. При этом способе используются следующие правила:

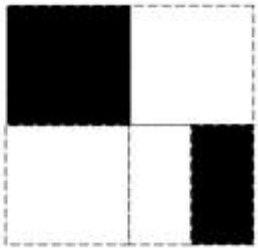
- Если изображение целиком белое, то оно представляется квадродеревом из одной «белой» вершины. Линейная запись такого квадродерева: 1.
- Если изображение целиком чёрное, то оно представляется квадродеревом из одной «чёрной» вершины. Линейная запись такого квадродерева: 0.
- Если на изображении есть и чёрные, и белые участки, то оно делится на 4 равные части (верхнюю левую, верхнюю правую, нижнюю левую, нижнюю правую) и представляется квадродеревом, состоящим из корневой вершины и четырёх поддеревьев, которые описывают части изображения. Пусть линейные записи поддеревьев таковы: <верхлевдерево>, <верхправдерево>, <нижнлевдерево>, <нижнправдерево>; тогда запись всего дерева будет такой:

T<верхлевдерево><верхпправдерево><нижнлевдерево><нижнправдерево>

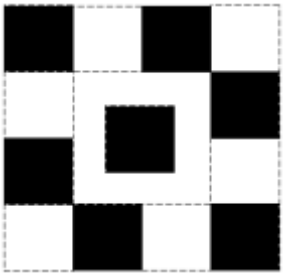
Пример: линейная запись квадродерева: T011T1010

квадродерево в виде графа: описываемое изображение:





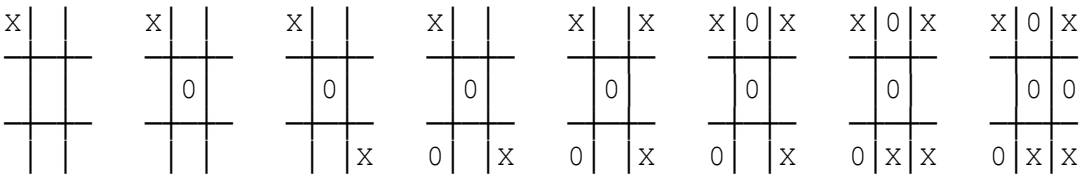
Для заданного ниже изображения составьте линейную запись квадродерева.



Задача 3. Крестики-нолики.

Предложите свой способ записи при помощи нулей и единиц информации о ходах партии игры в «Крестики-нолики». Предложенный способ должен быть как можно короче. Использовать в записи другие символы кроме 0 и 1 запрещено.

С помощью Вашего способа запишите течение указанной партии:



Сведения об игре:

Игра ведётся на квадратном поле 3 на 3 клетки. Игроки по очереди делают ходы, ставя свой знак в одну из пустых клеток. В начале игры поле пусто. Первым всегда делает ход игрок "крестик". Если одному из игроков удаётся выстроить в ряд 3 своих знака по вертикали, горизонтали или диагонали, то игра завершается его победой. Если все клетки поля заняты, но ни один из игроков не победил, то игра завершается ничьей.

Задача 4. Счетные палочки.

В древней восточной игре для подсчёта выигранных очков использовались специальные палочки. Игрок, заработавший какое-то количество очков, получал одну или несколько палочек, представляющих выигранную сумму очков. Использовались 5 типов палочек. Каждая палочка 1-го типа обозначала 2 очка, 2-го типа – 10 очков, 3-го типа – 50 очков, 4-го типа – 500 очков, 5-го типа 5000 очков. Чтобы узнать количество очков, представляемое набором палочек, нужно было сложить

номиналы всех палочек набора. Набор, в котором не было ни одной палочки, обозначал нулевое количество очков.

В конце игры игроки сравнивали наборы выигранных ими палочек. Чей набор обозначал большее количество очков, тот и считался победителем. Напишите программу, помогающую игрокам определять победителя.

На вход программа принимает две записи наборов счётных палочек всех игроков. Каждый набор записывается на отдельной строке последовательностью цифр через пробел. Каждая цифра обозначает одну палочку определённого типа: 1 – первого, 2 – второго ..., 5 – пятого. Цифры в последовательности могут повторяться и идти в любом порядке. Длина последовательности не более 100 000. Последовательность может быть пустой, если игрок не заработал ни одной палочки.

Программа выводит номер игрока, заработавшего максимальное количество очков. Если оба игрока заработали одинаковое количество очков, программа выводит 0.

Пример ввода:

```
4 1 1 2 1 1 1 4

2 4 2 4
```

Пример вывода:

```
0
```

Задания заключительного этапа олимпиады школьников «Ломоносов-2015» по информатике (10–11 классы)

Участники сдавали задания в тестирующую систему Ejudge.

Задача 1. Счётные палочки

В древней восточной игре для подсчёта выигранных очков использовались специальные палочки. Игрок, заработавший какое-то количество очков, получал одну или несколько палочек, представляющих выигранную сумму очков. Использовались 10 типов палочек. В зависимости от типа палочка имела разный номинал – обозначаемое количество очков. Чтобы узнать количество очков, представляемое набором палочек, нужно было сложить номиналы всех палочек набора. Набор, в котором не было ни одной палочки, обозначал нулевое количество очков.

В конце игры игроки сравнивали наборы выигранных ими палочек. Чей набор обозначал большее количество очков, тот и считался победителем. Напишите программу, помогающую игрокам определять победителя.

На вход программа принимает натуральное число N – количество игроков. Затем следуют десять натуральных чисел N0, N1, N2, ... N9, разделённых пробелами, – номиналы палочек каждого типа. Номиналы разных типов палочек не могут совпадать. Затем следуют записи наборов счётных палочек всех игроков. Каждый набор записывается на отдельной строке последовательностью цифр. Каждая цифра обозначает одну палочку определённого типа: 0 – нулевого, 1 – первого, ..., 9 – девятого. Цифры в последовательности могут повторяться и идти в любом порядке. Длина последовательности не более 100 000, номинал палочки не более 10 000. Последовательность может быть пустой, если игрок не заработал ни одной палочки.

Программа выводит номера игроков, заработавших максимальное количество очков. Номера должны выводиться в порядке возрастания.

Пример ввода:

```
2

1 2 3 4 5 6 7 8 9 10

00900

66
```

Пример вывода:

```
1 2
```

Задача 2. Квадродерево

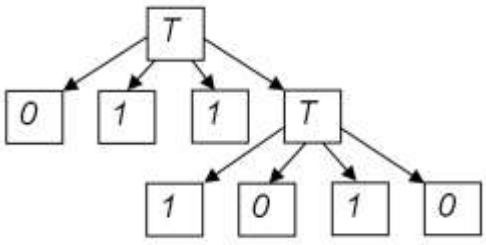
Рассмотрим способ представления растровых черно-белых квадратных изображений, называемый квадродеревом. При этом способе используются следующие правила:

- Если изображение целиком белое, то оно представляется квадродеревом из одной «белой» вершины. Линейная запись такого квадродерева: 1.
- Если изображение целиком чёрное, то оно представляется квадродеревом из одной «чёрной» вершины. Линейная запись такого квадродерева: 0.
- Если на изображении есть и чёрные, и белые участки, то оно делится на 4 равные части (верхнюю левую, верхнюю правую, нижнюю левую, нижнюю правую) и представляется квадродеревом, состоящим из корневой вершины и четырёх поддеревьев, которые описывают части изображения. Пусть линейные записи поддеревьев таковы: <верхлевдерево>, <верхправдерево>, <нижнлевдерево>, <нижнправдерево>; тогда запись всего дерева будет такой:

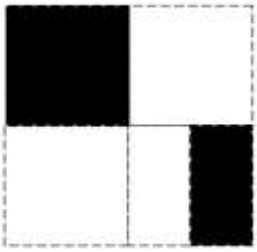
T<верхлевдерево><верхпправдерево><нижнлевдерево><нижнправдерево>

Пример: линейная запись квадродерева: T011T1010

квадродерево в виде графа:



описываемое изображение:



Для двух линейных записей квадродерева, доступных по ссылке http://ejudge.cs.msu.ru/kvadro/1, определите суммарную площадь белых участков изображений, представленных данными квадродеревьями. Считайте, что площадь самого мелкого квадратного белого участка равна 1.

При ответе в первой строке запишите два числа – ответ для первого дерева и ответ для второго. В последующих строках дайте обоснование правильности ответа (обоснованием может быть программа или словесное описание). Если для какого-то дерева ответ получен не был, то запишите в результат для него -1.

Примечание: в других вариантах предлагались другие квадродеревья и цвет участков.

Задача 3. Крестики-нолики

Игра ведётся на квадратном поле 3 на 3 клетки. Игроки по очереди делают ходы, ставя свой знак в одну из пустых клеток. В начале игры поле пусто. Первым всегда делает ход игрок «крестик». Если одному из игроков удаётся выстроить в ряд 3 своих знака по вертикали, горизонтали или диагонали, то игра завершается его победой. Если все клетки поля заняты, но ни один из игроков не победил, то игра завершается ничьей.

Предложите свой способ записи с помощью нулей и единиц информации о ходах партии игры в «Крестики-нолики». Предложенный способ должен быть как можно короче. Использовать в записи другие символы кроме 0 и 1 запрещено.

На основе Вашего способа реализуйте две функции:

1) функцию from_game_to_01, считывающую со стандартного потока ввода текстовую запись партии и переводящую её в нули и единицы;
2) функцию from_01_to_game, переводящую нули и единицы в текстовую запись партии и выводящую её на стандартный поток вывода.

Текстовая запись партии является последовательностью фрагментов, описывающих игровое поле после очередного хода. Фрагменты разделены пустыми строками. После последнего фрагмента пустой строки нет. Каждый фрагмент состоит из трёх строк, описывающих три горизонтали игрового поля. В каждой строке подряд идут описания трёх клеток слева направо. X (латинская заглавная буква) описывает клетку с крестом. 0 описывает клетку с нулём. # описывает пустую клетку.

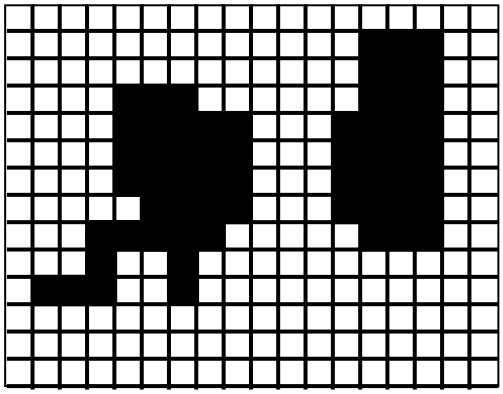
Пример текстовой записи:

```
X##
###
###

X##
###
##0

X##
#X#
##0
```

и т. д.

Задача 4. Квадрокоптер

В некотором плоском клетчатом прямоугольном мире (N на M клеток) существует 2 континента. Континент представляется связанным набором клеток. Континенты не имеют общих точек. Черные клетки (обозначаются 1) – суша, белые клетки (обозначаются 0) – вода.



На одном из них живет красный квадрокоптер, который мечтает съездить в путешествие на другой континент. Квадрокоптер умеет двигаться вверх, вниз, вправо, влево (соответствующие команды U, D, R, L), хранить в памяти любой кусочек карты и видеть в радиусе одной клетки (в том числе по диагонали). После перемещения на стандартный поток ввода будет записано то, что видит квадрокоптер вокруг себя. У квадрокоптера зарядки хватает на 8 действий движения. После истощения зарядки он приземляется на клетку, над которой находится. Если это суша – он может сесть и зарядиться от солнечной энергии. Если это вода, квадрокоптер тонет. Принудительная посадка квадрокоптера – P. Утверждается, что между 2 континентами существует путь, занимающий не более 5 действий квадрокоптера. Ваша задача перебраться с одного континента на другой. Если вы считаете, что квадрокоптер на другом континенте, нужно вывести W.

В начале работы программы на стандартном потоке ввода находится информация о том, что видит квадрокоптер вокруг себя.

Пример протокола взаимодействия квадрокоптера:

Стандартный поток вывода	**Стандартный поток ввода**
	0 0 0 0 1 0 0 0 0
R	0 0 0 1 0 0 0 0 0
R	0 0 0 0 0 1 0 0 0

R	0 0 1 0 1 1 0 0 0
P	0 0 1 0 1 1 0 0 0
W	

Указание: после каждого вывода требуется произвести операцию очистки буфера вывода. Примеры кода для этого действия можно найти на http://ejudge.cs.msu.ru/kvadrocopter.

Задача 5. Анализ программы

Посмотрите на функцию. У нее есть аргументы. При разных значениях аргументов функция может завершиться и возвратить то или другое значение или может не завершиться (зациклиться или совершить некорректную операцию). Определите, сколько различных чисел может возвратить эта функция, если она завершилась. Например, следующая функция возвращает 2 различных числа (10 и 20):

```
int ggg(int x, int y) {
    if (x < y) {
       return 10;
    } else {
       return 20;
    }
}
```

Предполагайте, что функция выполняется на компьютере, где типы данных имеют следующие ограничения: int - от -2^{31} до $2^{31}-1$.

В ответе укажите на первой строке искомое количество, а, начиная со второй строки, кратко опишите способ его получения.

```
int quest(int x) {
    while (x < 1024) {
       for(int y = 0; y < x; y++) {
          if (y * y == x) {
             return y;
          }
```

```
        }

        x ++;
    }
    return -1;
}
```

Задача 6. Черный ящик

Вам удалось получить доступ к программе, которая работает с последовательностью различных латинских символов. Ваша задача – проанализировать то, что делает программа, и составить текстовое описание её работы. Опишите процесс получения результата.

В предположении, что программа выдала результат ‘acbzfde’, укажите, что было подано на вход программе.

Исполняемый файл можно скачать по ссылке http://ejudge.cs.msu.ru/blackbox/1.exe. Ввод данных осуществляется из файла input.txt, вывод в файл output.txt.

В первой строке ответа запишите ввод для заданного результата, а в последующих строках – текстовое описание работы и обоснование.

Указание: в задаче можно считать, что «черный ящик» корректно работает только на последовательностях различных строчных латинских символов.

Примечание: в других вариантах предлагались другие черные ящики.

Ответы отборочного этапа олимпиады школьников «Ломоносов» по информатике 2014/2015 учебного года

5-9 классы

Задача 1. Системы счисления

Ответ: 23.

Задачу можно решить при помощи стандартной программы-калькулятора, входящей практически во все современные операционные системы для персональных компьютеров.

Задача 2. Шахматы

Ответ: c – a = d – b OR c – a = b – d OR a – c = d – b OR a – c = b – d OR a = c AND -1 <= b – d AND b – d <= 1 OR b = d AND -1 <= a – c AND a – c <= 1.

Фигура-слон бьет все фигуры, находящиеся по диагонали от себя. Фигура-король бьет все фигуры, находящиеся на расстоянии 1 хода от нее (по диагонали, вертикали или горизонтали).

Задача 3. Робот

22

Задача 4. Шифр

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practically it beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one and preferably only one obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than right now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea let's do more of those!

Задача 5. Малыш и Карлсон

2 или 3 (в зависимости от учета ведущего нуля)

Задача 6. Бармаглот-1

?](1~-]]Z.0<)"

Задача 7. Бармаглот-2

0]?(+#]?)[_}0](1+.]]@.{}-]0=!)"

Задача 8. Системы счисления – 2

39827

10-11 классы. Первый тур

Задача 1. Упаковка числа (Варианты 1,3,5,7) Рассмотрим способ упакованной записи чисел в двоичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 1111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в двоичной системе и закрывающей скобки –]. Так число 111111100 в упакованной записи будет записано следующим образом: 1[1[1[10]]]0[10] Найдите наименьшую по длине упакованную запись двоичного числа :

1. 11100000000100000111111111110
3. 11111111111000111000000000100000
5. 11000001111111111100011100000000 1
7. 11000000001110000001111111111110001

Задача будет оценена в зависимости от верности упаковки и длины получившейся последовательности.

Задача 1. Упаковка числа (Варианты 2,4,6,8) Рассмотрим способ упакованной записи чисел в троичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 111111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в троичной системе и закрывающей скобки –]. Так число 11111111000 в упакованной записи будет записано следующим образом: 1[2[2]]0[10] Найдите наименьшую по длине упакованную запись троичного числа:

2. 11122222222100001111111111110
4. 11111111111100111222222210000
6. 22222222100011111111111001110
8. 11102222222210001111111111100

Задача будет оценена в зависимости от верности упаковки и длины получившейся последовательности.

Ответы:

1	1110[1000]1000001[1011]0	24
2	1112[22]100001[111]0	20
3	1[1011]0001110[1000]100000	26

4	1[111]001112[22]10000	21
5	11000001[1011]0001110[1000]1	28
6	2[22]10001[111]001110	21
7	110[1000]111000001[1011]0001	28
8	11102[22]10001[111]00	21

Задача 2. Последовательность

1. Последовательность чисел 38, 37, 36, 35, 39, 34, 33, 32, 31, 30, ... , 100 была получена выписыванием 500 первых натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в обратном алфавитном порядке. Определите число, стоящее в последовательности на 114-м месте.
2. Последовательность чисел 38, 37, 36, 35, 39, 34, 33, 32, 31, 30, ... , 100 была получена выписыванием 500 первых натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в обратном алфавитном порядке. Определите номер места, на котором стоит в последовательности число 114.
3. Последовательность чисел 100, 200, 300, 301, 302, 303, 304, 309, 350, 351 ... 38 была получена выписыванием 500 первых натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в алфавитном порядке. Определите число, стоящее в последовательности на 114-м месте.
4. Последовательность чисел 100, 200, 300, 301, 302, 303, 304, 309, 350, 351 ... 38 была получена выписыванием 500 первых натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в алфавитном порядке. Определите номер места, на котором стоит в последовательности число 114.
5. Последовательность чисел 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, ... , 100 была получена выписыванием 500 первых чётных натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в обратном алфавитном порядке. Определите число, стоящее в последовательности на 114-м месте.
6. Последовательность чисел 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, ... , 100 была получена выписыванием 500 первых чётных натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в обратном алфавитном порядке. Определите номер места, на котором стоит в последовательности число 114.
7. Последовательность чисел 100, 200, 300, 302, 304, 350, 352, 354, 356, 358, ... , 38 была получена выписыванием 500 первых чётных натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в алфавитном порядке. Определите номер места, на котором стоит в последовательности число 114.
8. Последовательность чисел 38, 37, 36, 35, 39, 34, 33, 32, 31, 30, ... , 100 была получена выписыванием 500 первых натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в обратном алфавитном порядке. Определите число, стоящее в последовательности на 114-м месте.

9. Последовательность чисел 37, 35, 39, 33, 31, 27, 25, 29, 23, 21, ... , 301 была получена выписыванием 500 первых нечётных натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в обратном алфавитном порядке. Определите число, стоящее в последовательности на 411-м месте.
10. Последовательность чисел 37, 35, 39, 33, 31, 27, 25, 29, 23, 21, ... , 301 была получена выписыванием 500 первых нечётных натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в обратном алфавитном порядке. Определите номер места, на котором стоит в последовательности число 411.
11. Последовательность чисел 301, 303, 309, 351, 353, 359, 355, 357, 361, 363, ... , 37 была получена выписыванием 500 первых нечётных натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в алфавитном порядке. Определите число, стоящее в последовательности на 411-м месте.
12. Последовательность чисел 301, 303, 309, 351, 353, 359, 355, 357, 361, 363, ... , 37 была получена выписыванием 500 первых нечётных натуральных чисел так, чтобы их записи римскими цифрами были отсортированы в алфавитном порядке. Определите номер места, на котором стоит в последовательности число 411.

Ответы:

1	2	3	4	5	6	7	8	9	10	11	12
125	**136**	**255**	**365**	**644**	**268**	**233**	**125**	**217**	**370**	**569**	**131**

Задача 3.

Возможны другие правильные ответы.

Вариант	Возможный ответ
1	INV(MASK(TRANS(MASK(MAX(X, INV(X))))))
2	INV(TRANS(MASK(INV(DIF(X, X)))))
3	DIF(TRANS(MASK(INV(DIF(X, X)))), MASK(INV(DIF(X, X))))
4	DIF(TRANS(MASK(DIF(X, X))), MASK(DIF(X, X)))
5	INV(MIN(MASK(TRANS(INV(MASK(MIN(INV(X),X))))), TRANS(MASK(TRANS(INV(MASK(MIN(INV(X),X))))))))

6	MAX(TRANS(MASK(MIN(X, INV(X)))) MASK(MIN(X, INV(X))))
7	MASK(TRANS(MASK(MIN(A, INV(A))))

Задача 4.

В задаче требуется написать программу. При запаковке следовало учитывать, что короткие последовательности не требуется изменять.

Задача 5.

Вариант	Ответ
1	punycode-был-разработан-для-однозначного-преобразования-доменных-имён-в-последовательность-ascii-символов
2	punycode--стандартизированный-метод-преобразования-последовательностей-unicode-символов
3	punycode-позволяет-конвертировать-набор-символов-в-кодировке-unicode-в-набор-символов-для-существующей-dns
4	punycode--это-способ-приведения-интернационализированных-доменных-имен-содержащих-в-себе-unicode-символы

Задача 6.

Вариант	Ответ
1	221111
2	12122112212122
3	22121121121211

4	11212212212122

Задача 7.

Вариант	Ответ
1	333359 5/6 333353 5/6 333286 1/3
2	333692 5/6 333298 5/6 333008 1/3
3	333109 1/3 333335 5/6 333554 5/6
4	333386 5/6 333388 1/3 333224 5/6
5	333382 333168 333450
6	333233 1/3 333619 1/3 333147 1/3
7	333555 5/6

	332897 1/3 333546 5/6
8	333861 1/2 333015 333123 1/2

Задача 8.

```
0]0][?(]?)[}[#]{+}]4~/]4*]{}=(0]{=([}[#]{+}]4~}/]4*]{=(Y.E.S.")N.O.")
Y.E.S.")N.O."
```

Задача 9.

```
0]0]?((+#]?]![[@]1!)[]]@=([!])[)[_}[]]{}>([{]}1!){0]?)[[."
```

Задача 10.

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <string>
#include <cstring>
#include <cctype>

using namespace std;

const char *parse_string = "([920, 438]7(27)8([0, 0]14(39)16(30)-
1)11(49)12(48)13([56, 536]13([86, 856]11([176, 526]13(2)14([186,
96]14([196, 106]13(35)14([216, 156]17(31)18([216, 396]11(4)12([246,
906]8([266, 446]13([316, 976]15(41)16([326, 646]14([336, 216]11([336,
686]10(38)11([396, 876]10(5)11([426, 0]12(44)13([426,
966]17(9)18([436, 956]10(36)11([446, 806]12(18)13([466, 796]13([476,
396]16([476, 556]13(10)14([546, 226]12([626, 456]14(40)15([636,
86]12([686, 856]7(19)8([716, 706]11(23)12([726, 216]15(3)16([736,
196]10([746, 766]13([766, 726]12([796, 886]13([816,
356]16(37)17([856, 926]13([886, 626]11(20)12([926, 96]11([946,
```

```
636]8(11)9([956, 616]12([956, 816]14(15)15(34)-1)13(32)-1)-1)12(33)-
1)-1)14(22)-1)-1)14(43)-1)13(16)-1)14(46)-1)11(14)-1)-1)-1)-1)13(21)-
1)-1)13(26)-1)-1)17(17)-1)14(28)-1)-1)-1)-1)-1)-1)-1)12(50)-1)15(7)-
1)-1)14(45)-1)9(6)-1)-1)-1)-1)15(13)-1)-1)12(29)-1)14(25)-
1)14(47)15(8)16(1)17(24)18(42)19(12)-1)";


struct Tree

{

   int w = -1;

   int h = -1;

   vector< pair< int, Tree * > > sons;

   friend std::ostream&operator<<(std::ostream &out, const Tree
&tree)

   {

       out << "(";

       if (tree.w != -1) {

           out << "[" << tree.w << ", " << tree.h << "]";

           for (auto nxt : tree.sons) {

               out << nxt.first << *(nxt.second);

           }

           out << "-1)";

       } else {

           out << tree.h << ")";

       }

       return out;

   }

   int height() const

   {

       int res = 0;

       for (auto nxt : sons) {

           int val = nxt.second->height() + 1;

           if (res < val) {
```

```
                res = val;
            }
        }
        return res;
    }
    ~Tree()
    {
        for (auto nxt : sons) {
            delete nxt.second;
        }
    }
};

int
getValue(const char **s)
{
    while (**s != '-' && !isdigit(**s)) {
        ++(*s);
    }
    int res = 0;
    bool sign = false;
    if (**s == '-') {
        sign = true;
        ++(*s);
    }
    while (isdigit(**s)) {
        res = res * 10 + **s - '0';
        ++(*s);
    }
    ++(*s);
```

```
    return sign ? -res : res;
}

Tree *
construct(const char **s)
{
    Tree *res = new Tree();
    const char *ptr = *s;
    if (**s == '[') {
        res->w = getValue(&ptr);
        res->h = getValue(&ptr);
        if (res->w == -1 || res->h == -1) {
            cerr << "fail!" << endl;
            cerr << *s << endl;
        }
        int tmp = getValue(&ptr);
        while (tmp != -1) {
            Tree *new_son = construct(&ptr);
            res->sons.push_back(make_pair(tmp, new_son));
            tmp = getValue(&ptr);
        }
    } else { // List
        res->h = getValue(&ptr);
    }
    *s = ptr;
    return res;
}

int
main()
```

```
{
    const char *s1 = parse_string + 1;
    Tree *decision_tree = construct(&s1);
    while (decision_tree->w != -1) {
        int curX = decision_tree->w, curY = decision_tree->h;
        cout << "? " << curX << " " << curY << endl;
        int sum;
        cin >> sum;
        bool findVal = false;
        for (auto x : decision_tree->sons) {
            if (x.first == sum) {
                findVal = true;
                decision_tree = x.second;
                break;
            }
        }
        if (!findVal) {
            cerr << "Oops!" << endl;
            return 1;
        }
    }
    cout << "! " << decision_tree->h << endl;
    return 0;
}
```

10-11 классы. Второй тур

Задача 1. Восстановите связное сообщение по тексту, который был получен после некоторого преобразования исходного текста.

Вариант	Ответ
1	A numeric character reference in HTML refers to a character by its Universal Character Set/Unicode code point.
2	Web pages authored using HTML may contain multilingual text represented with the Unicode universal character set.
3	The relationship between Unicode and HTML tends to be a difficult topic for many computer professionals.
4	It is possible to represent characters from the whole of Unicode inside an HTML document by using a numeric character reference.

Задача 2. Последовательность.

Вариант:	1	2	3	4
Ответ:	**297**	**2016**	**134**	**555**

Задача 3.

Возможное решение.

```
import sys


def main():
    number_of_view = 0
    number_of_sharing = 0
```

```
    for line in sys.stdin:
        date, time, action = line.split()
        if date < '2014-12-31' or
           date == '2014-12-31' and time < '07:00:00':
            if action == 'W':
                number_of_view += 1
            elif action == 'S':
                number_of_sharing += 1
            else:
                pass

    christmas_tree_size = number_of_view + 5 * number_of_sharing

    fibonacci = [1, 1]
    sum_fibonacci = [0, 1]

    while sum_fibonacci[-1] < christmas_tree_size:
        fibonacci.append(fibonacci[-1] + fibonacci[-2])
        sum_fibonacci.append(fibonacci[-1] + sum_fibonacci[-1])

    length = 100 + (len(sum_fibonacci) - 2) * 20
    print(length)

main()
```

Задача 4.

```
#include <cstdio>
#include <cstdlib>
#include <cmath>
```

```
#include <cstring>
#include <ctime>
#include <cassert>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <set>
#include <map>
#include <algorithm>
#include <string>
#include <sstream>

using namespace std;

int getnum(const string &s, int &i) {
    int ans = 0;
    int lastd = -1;

    for (; i < s.length(); i++) {
        if (s[i] != '[' && lastd != -1) {
            ans *= 2;
            ans += lastd;
        }

        if (s[i] == '[') {
            i++;
            int num = getnum(s, i);
            for (int j = 0; j < num; j++) {
                ans *= 2;
```

```
                ans += lastd;
            }
            lastd = -1;
            continue;
        }

        if (s[i] == ']') {
            return ans;
        }
        lastd = s[i] - '0';
    }
}

string convert(const string& s) {
    string ans = "";
    for (int i = 0; i < s.length(); i++) {
        if (i + 1 == s.length()) {
            ans += s[i];
            continue;
        }
        if (s[i + 1] == '[') {
            char c = s[i];
            i += 2;
            int num = getnum(s, i);
            for (int j = 0; j < num; j++) {
                ans += c;
            }
        } else {
            ans += s[i];
        }
```

```
    }
    return ans;
}

bool lessnumber(const string & a, const string & b) {
    if (a.length() != b.length()) return a.length() < b.length();
    return a < b;
}

int main()
{
    string s, t;
    cin >> s >> t;

    string ss = convert(s), tt = convert(t);

    if (lessnumber(ss, tt)) {
        cout << "<\n";
    } else if (ss == tt) {
        cout << "=\n";
    } else {
        cout << ">\n";
    }

    return 0;
}
```

Задача 5.

В задаче может быть несколько правильных ответов.

Задача 6.

111111222212212122

Задача 7.

Вариант 1.

Balan 34

Cynric 34

Arthur 32

Geraint 32

Gingalain 32

Josephus 32

Dinadan 30

Mordred 30

Bruin 28

Calogrenant 28

Lanval 28

Owain 28

Rience 28

Claudin 26

Culhwch 26

Elyan 26

Epinogres 26

Maleagant 26

Mark 26

Meliodas 26

Palamedes 26

Ysbaddaden 26

Aglovale 24

Balin 24

Ban 24

Gaheris 24

Galahad 24

Gareth 24

Gorlois 24

Griflet 24

Laudine 24

Lionel 24

Daniel 22

Edern 22

Galeschin 22

Lamorak 22

Lot 22

Morien 22

Pelleas 22

Pelles 22

Tom 22

Vortimer 22

Bedivere 20

Brutus 20

Ector 20

Esclados 20

Galehault 20

Gwyn 20

Hengest 20

Horsa 20

Lunete 20

Morvydd 20

Pellinore 20

Sagramore 20

Dagonet 18

Eliwlod 18

Erec 18

Faerie 18

Gawain 18

Kahedin 18

Leodegrance 18

Lucan 18

Meirchion 18

Oberon 18

Safir 18

Breunor 16

Cador 16

Cerdic 16

Claudas 16

Constantine 16

Feirefiz 16

Fisher 16

Hueil 16

Loholt 16

Merlin 16

Segwarides 16

Tor 16

Ywain 16

Amr 14

Bagdemagus 14

Caradoc 14

Constans 14

Esclabor 14

Gornemant 14

Joseph 14

Kay 14

Lucius 14

Mabon 14

Melehan 14

Morgan 14

Taliesin 14

Brangaine 12

Durnure 12

Modron 12

Manawydan 10

Percival 10

Vortigern 10

Catigern 8

Menw 8

Вариант 2.

Galehault 38

Gwyn 38

Hueil 36

Arthur 30

Balin 30

Gornemant 30

Josephus 30

Manawydan 30

Sagramore 30

Segwarides 30

Cerdic 28

Constantine 28

Feirefiz 28

Geraint 28

Hector 28

Lucan 28

Merlin 28

Vortigern 28

Ywain 28

Amr 26

Brangaine 26

Brutus 26

Esclados 26

Gareth 26

Horsa 26

Lunete 26

Rience 26

Aurelius 24

Blanchefleur 24

Esclabor 24

Lionel 24

Morien 24

Percival 24

Agravain 22

Bagdemagus 22

Bedivere 22

Catigern 22

Lamorak 22

Lucius 22

Maleagant 22

Modron 22

Mordred 22

Morgan 22

Pelleas 22

Balan 20

Claudin 20

Edern 20

Eliwlod 20

Guiron 20

Hoel 20

Kahedin 20

Lancelot 20

Ysbaddaden 20

Daniel 18

Dinadan 18

Dindrane 18

Gaheris 18

Galahad 18

Gorlois 18

Joseph 18

Madoc 18

Meliodas 18

Pellam 18

Tom 18

Breunor 16

Cador 16

Meirchion 16

Pellinore 16

Aglovale 14

Culhwch 14

Dagonet 14

Epinogres 14

Erec 14

Faerie 14

Gawain 14

Laudine 14

Leodegrance 14

Lot 14

Morholt 14

Oberon 14

Owain 14

Pelles 14

Taliesin 14

Urien 14

Vortimer 14

Bruin 12

Claudas 12

Constans 12

Fisher 12

Loholt 12

Mabon 12

Palamedes 12

Tor 12

Ector 10

Elyan 10

Hengest 10

Tristan 10

Durnure 8

Melehan 8

Вариант 3.

Gingalain 42

Brutus 34

Kahedin 34

Lamorak 30

Leodegrance 30

Pelleas 30

Catigern 28

Edern 28

Gorlois 28

Joseph 28

Lancelot 28

Mordred 28

Percival 28

Arthur 26

Ban 26

Caradoc 26

Cerdic 26

Dindrane 26

Elyan 26

Lanval 26

Merlin 26

Pellam 26

Rience 26

Safir 26

Tristan 26

Eliwlod 24

Epinogres 24

Esclados 24

Guiron 24

Hueil 24

Kay 24

Loholt 24

Madoc 24

Melehan 24

Morgan 24

Palamedes 24

Blanchefleur 22

Calogrenant 22

Claudin 22

Durnure 22

Esclabor 22

Hector 22

Maleagant 22

Aurelius 20

Bagdemagus 20

Galahad 20

Gornemant 20

Lucius 20

Meirchion 20

Owain 20

Pelles 20

Tom 20

Tor 20

Urien 20

Balan 18

Bruin 18

Cador 18

Constantine 18

Culhwch 18

Dagonet 18

Gaheris 18

Gawain 18

Josephus 18

Laudine 18

Lionel 18

Manawydan 18

Taliesin 18

Ysbaddaden 18

Brangaine 16

Claudas 16

Constans 16

Daniel 16

Galehault 16

Horsa 16

Mabon 16

Menw 16

Morholt 16

Dinadan 14

Faerie 14

Hoel 14

Morien 14

Oberon 14

Segwarides 14

Aglovale 12

Breunor 12

Galeschin 12

Gareth 12

Geraint 12

Lot 12

Lucan 12

Lunete 12

Mark 12

Meliodas 12

Modron 12

Morvydd 10

Amr 8

Balin 8

Ector 8

Hengest 8

Вариант 4.

Edern 38

Pellam 34

Ban 32

Geraint 30

Lionel 30

Brangaine 28

Elyan 28

Erec 28

Fisher 28

Guiron 28

Breunor 26

Bruin 26

Culhwch 26

Durnure 26

Galahad 26

Gorlois 26

Laudine 26

Segwarides 26

Arthur 24

Aurelius 24

Bagdemagus 24

Dagonet 24

Dinadan 24

Esclados 24

Gaheris 24

Gareth 24

Griflet 24

Lanval 24

Madoc 24

Mark 24

Modron 24

Palamedes 24

Tor 24

Vortigern 24

Agravain 22

Calogrenant 22

Catigern 22

Constantine 22

Faerie 22

Hoel 22

Hueil 22

Loholt 22

Lunete 22

Morien 22

Ysbaddaden 22

Bedivere 20

Brutus 20

Gingalain 20

Gornemant 20

Mabon 20

Menw 20

Morvydd 20

Pelleas 20

Taliesin 20

Tom 20

Ywain 20

Aglovale 18

Claudas 18

Ector 18

Feirefiz 18

Gwyn 18

Kahedin 18

Lancelot 18

Lucan 18

Lucius 18

Manawydan 18

Merlin 18

Mordred 18

Pelles 18

Sagramore 18

Caradoc 16

Claudin 16

Constans 16

Daniel 16

Dindrane 16

Eliwlod 16

Hengest 16

Joseph 16

Kay 16

Lot 16

Meliodas 16

Rience 16

```
Vortimer 16
Cador 14
Galehault 14
Gawain 14
Horsa 14
Melehan 14
Morgan 14
Morholt 14
Epinogres 12
Hector 12
Owain 12
Galeschin 10
Cynric 8
Esclabor 8
Josephus 8
Urien 8
Pellinore 6
```

Задача 8.

Возможное решение.

```
?](?])[[.[(.[)"
```

Задача 9.

Возможное решение.

```
?}(1](1+]]}{}%]0=!]{]]{}=!}[}{}{&]{[){=(Y.1]!)[!(N.!)?})"
```

Задача 10.

```
#include <iostream>
```

```
#include <algorithm>

#include <vector>

#include <string>

#include <cstring>

#include <cctype>


using namespace std;


const char *parse_string = "([2, 124]0([0, 2]0([0, 0]0([0, 5]0([0,
8]0(42)1(1)-1)1(24)-1)1([0, 1]0(47)1(39)-1)-1)1([0, 3]0([0,
7]0(12)1(8)-1)1([0, 4]0(27)1([0, 5]0(49)1(30)-1)-1)-1)-1)1([56,
539]0([86, 858]0([176, 528]0(2)1([186, 97]0([196, 108]0(35)1([216,
162]0(31)1([216, 401]0([247, 908]0(6)1([268, 450]0(45)1([318,
979]0(41)1([326, 647]0([336, 688]0(38)1([337, 217]0([396,
877]0(5)1([426, 0]0(44)1([426, 974]0([436, 957]0(36)1([446,
806]0(18)1([466, 802]0(28)1([476, 398]0([476, 561]0([546, 226]0([626,
457]0(40)1([636, 86]0([686, 860]0(19)1([716, 708]0(23)1([726,
221]0([736, 197]0([746, 770]0([766, 726]0([796, 887]0([816,
361]0([856, 927]0([886, 628]0(20)1([916, 436]0(48)1([926, 99]0([946,
638]0(11)1([956, 616]0([957, 821]0(34)1(15)-1)1(32)-1)-1)1(33)-1)-1)-
1)1(22)-1)1(37)-1)1(43)-1)1(16)-1)1(46)-1)1(14)-1)1(3)-1)-1)-1)1(21)-
1)-1)1(26)-1)1(10)-1)1(17)-1)-1)-1)-1)1(9)-1)-1)-1)1(50)-1)-1)1(7)-
1)-1)-1)-1)1(4)-1)-1)-1)1(13)-1)-1)1(29)-1)1(25)-1)-1)";


struct Tree

{

   int w = -1;

   int h = -1;

   vector< pair< int, Tree * > > sons;

   friend std::ostream&operator<<(std::ostream &out, const Tree
&tree)

   {

       out << "(";

       if (tree.w != -1) {

           out << "[" << tree.w << ", " << tree.h << "]";

           for (auto nxt : tree.sons) {

               out << nxt.first << *(nxt.second);
```

```
            }
            out << "-1)";
        } else {
            out << tree.h << ")";
        }
        return out;
    }
    int height() const
    {
        int res = 0;
        for (auto nxt : sons) {
            int val = nxt.second->height() + 1;
            if (res < val) {
                res = val;
            }
        }
        return res;
    }
    ~Tree()
    {
        for (auto nxt : sons) {
            delete nxt.second;
        }
    }
};

int
getValue(const char **s)
{
    while (**s != '-' && !isdigit(**s)) {
```

```
        ++(*s);
    }
    int res = 0;
    bool sign = false;
    if (**s == '-') {
        sign = true;
        ++(*s);
    }
    while (isdigit(**s)) {
        res = res * 10 + **s - '0';
        ++(*s);
    }
    ++(*s);
    return sign ? -res : res;
}

Tree *
construct(const char **s)
{
    Tree *res = new Tree();
    const char *ptr = *s;
    if (**s == '[') {
        res->w = getValue(&ptr);
        res->h = getValue(&ptr);
        if (res->w == -1 || res->h == -1) {
            cerr << "fail!" << endl;
            cerr << *s << endl;
        }
        int tmp = getValue(&ptr);
        while (tmp != -1) {
```

```
            Tree *new_son = construct(&ptr);
            res->sons.push_back(make_pair(tmp, new_son));
            tmp = getValue(&ptr);
        }
    } else { // List
        res->h = getValue(&ptr);
    }
    *s = ptr;
    return res;
}

int
main()
{
    const char *s1 = parse_string + 1;
    Tree *decision_tree = construct(&s1);
    while (decision_tree->w != -1) {
        int curX = decision_tree->w, curY = decision_tree->h;
        cout << "? " << curX << " " << curY << endl;
        int sum;
        cin >> sum;
        bool findVal = false;
        for (auto x : decision_tree->sons) {
            if (x.first == sum) {
                findVal = true;
                decision_tree = x.second;
                break;
            }
        }
        if (!findVal) {
```

```
            cerr << "Oops!" << endl;
            return 1;
        }
    }
    cout << "! " << decision_tree->h << endl;
    return 0;
}
```

Ответы и критерии оценивания заданий заключительного этапа олимпиады школьников "Ломоносов" по информатике 2014/2015 учебного года

5-9 классы

Задача 1. Системы счисления.

Быстрое решение:

Поскольку $3^3=27$, можно сразу переводить из троичной системы в 27-ричную. Для этого разобъём запись числа в троичной системе на тройки цифр, начиная с самой правой цифры:
2 211 201 122 110 201
Теперь каждую триаду переведём из троичной в десятичную систему счисления:
2 22 19 17 12 19
В 27-ричной системе 22 соответствует цифре M, 19 – J, 17 – H, 12 – C, следовательно ответ записывается так: $2MJHCJ_{27}$

Медленное решение:

Решать обычным способом, используя как промежуточную десятичную систему, медленно, неудобно и рискованно из-за возможных арифметических ошибок в расчётах. Всё же рассмотрим этом метод. Для перевода числа в десятичную систему воспользуемся схемой Горнера, позволяющей сэкономить на вычислениях:

$2211201122110201_3 =$

$=(2\times3+2)\times3+1)\times3+1)\times3+2)\times3+0)\times3+1)\times3+1)\times3+2)\times3+2)\times3+1)\times3+1)\times3+0)\times3+2)\times3+0)\times3+1$

$=40776229_{10}$

Переводим число из десятичной системы в 27-ричную. Для этого находим остаток и частное от деления нацело числа 40776229 на 27, далее повторяем деление для полученного частного и т. д. Получаем 19, 12, 17, 19, 22 и 2. Выписываем в обратном порядке цифрами 27-ричной системы: $2MJHCJ_{27}$

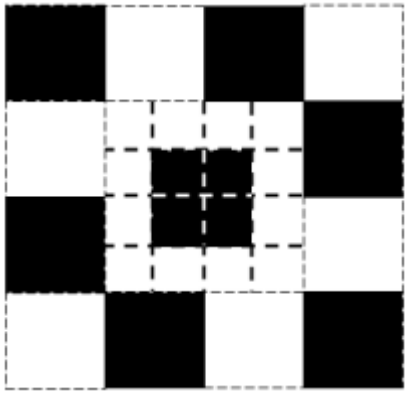
Ответ к задаче 1: $2MJHCJ_{27}$

Критерий к задаче 1: Полный балл ставился за верный ответ с правильным обоснованием.

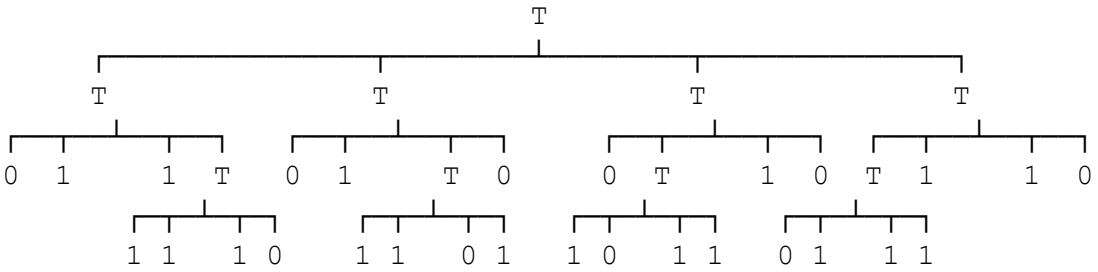
Задача 2. Квадродерево. Решение

Чтобы решить задачу, нужно разбить исходное изображение на квадраты по принципу, на котором основано квадродерево, затем построить квадродерево в виде графа, после чего составить линейную запись построенного дерева.

После разбиения изображение выглядит так, как показано на рисунке:



Граф квадродерева выглядит так:



Линейная запись всего квадродерева состоит из следующих частей:

T<запись 1-го поддерева><запись 2-го п/д><запись 3-го п/д><запись 4-го п/д>

Находим запись первого поддерева: T011T1110

Находим запись второго поддерева: T01T11010

Находим запись третьего поддерева: T0T101110

Находим запись четвертого поддерева: TT0111110

По найденным записям составляем запись ответа:

TT011T1110T01T11010T0T101110TT0111110

Ответ к задаче 2: TT011T1110T01T11010T0T101110TT0111110

Критерий к задаче 2: Полный балл ставился за верный ответ с правильным обоснованием.

Задача 3. Крестики-нолики. Решение

Рассмотрим различные способы записи партий, чтобы выбрать из них самый короткий, как этого требуется в условии задачи.

Первый очевидный способ строится по тому как партия представлена в условии. Там дана последовательность снимков игрового поля, сделанных после очередного хода. Поле состоит из девяти клеток. Каждая клетка может быт либо пуста, либо заполнена X, либо 0. Значит, для кодирования клетки можно использовать два бита: 00 – пустая клетка; 01 – клетка с X, 10 – клетка с 0. Снимок поля составляется из записей девяти клеток (например,

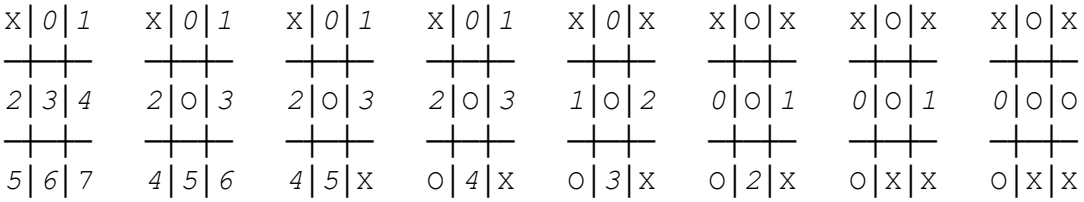
следующих по горизонталям от верхней к нижней, а внутри горизонтали слева направо).
Запись снимка поля после первого хода:010000000000000000 занимает 18 бит Запись партии из условия по первому способу:
010000000000000000010000001000000000010000001000000000101000000100010000101000100100010001011001001000100001011001001000100010101100100101010010100101
Она занимает 8 ×18 = 144 бит.
Попробуем найти более короткий способ. Заметим, что в записи по первому способу много места отводится под пустые клетки, и многократно дублируются сведения о ранее сделанных ходах. Что если записывать партию сделанными ходами? Каждый ход будем записывать, указывая клетку, в которую поставлен символ. Для этого занумеруем клетки:

```
0 | 1 | 2
---+---+---
3 | 4 | 5
---+---+---
6 | 7 | 8
```

Указывать, какой именно ставится символ необязательно, так как порядок, по которому делают ходы игроки, единый для всех партий. На первом ходу всегда ставится X, на втором – 0, на третьем – X и т. д. Ход будем кодировать номером клетки, записанным в двоичной системе. На каждый ход нам потребуется 4 бита, что много меньше, чем 18 бит для снимка поля. Запись партии по второму способу: 00000100100001100010000101110101 занимает 8 × 4 = 32 бита, что много меньше, чем в первом способе.
Можно ли улучшить второй способ? Если бы нам удалось не использовать восьмёрку, то каждый ход занимал бы 3 бита, что дало бы существенную экономию. Поступим так. Будем записывать ход в верхнюю левую клетку и ход в правую нижнюю клетку одним и тем же кодом: 000. Чтобы различать какой из этих двух ходов сделан в партии раньше, добавим дополнительный бит в конец записи. Ноль будет означать, что когда 000 встретится в записи в первый раз, это следует прочесть как ход в верхнюю левую клетку, а во второй раз – в нижнюю правую. Единица будет указывать на обратное. Запись партии по улучшенному второму способу: 000100000110010001111101 0 занимает 25 бит.
Можно ли записывать ещё короче? Заметим, что последнее усовершенствование использовало тот факт, что повторно сделать ход в уже занятую клетку нельзя. Значит, записывая ходы в конце партии, можно сэкономить за счёт указания номера клетки, но не одной из девяти клеток поля, а одной из оставшихся свободными клеток, которых меньше, чем 9. Чтобы нумеровать свободные клетки будем рассматривать клетки также как при предыдущей нумерации, но занятые клетки будем пропускать, не давая им номера. Например, после пятого хода в заданной партии останется четыре свободные клетки, занумерованные так (ходы второго игрока обозначены символом O, чтобы отличать от свободной клетки с номером *0*):

```
X | 0 | X
---+---+---
1 | O | 2
```


Таким образом, для первого хода будет затрачено четыре бита, для ходов со второго по пятый – три бита, для шестого и седьмого ходов – два бита, для восьмого хода – один бит. На девятом ходу остаётся единственная клетка, но ход либо будет сделан и партия закончится (0), либо не будет сделан (1). Значит, тоже нужен один бит. Запись заданной партии займёт 4 + 4 × 3 + 2 × 2 + 2 = 22 бита. Для удобства составления этой записи приведём нумерацию свободных клеток после первого, второго и других ходов (ходы второго игрока по-прежнему обозначены символом O, чтобы отличать от свободной клетки с номером *0*)


Полученная запись: 00000111101000010010 11

Ещё один способ заключается в том, чтобы перенумеровать все возможные партии и сделать записью партии её номер. Всего может быть 1 партия, в которой не сделано ни одного хода, 9 партий длиной в 1 ход, 9×8 партий длиной в 2 хода, 9×8×7 партий длиной в 3 хода, 9×8×7×6 партий длиной в 4 хода, 9×8×7×6×5 партий длиной в 5 ходов. Партий длиной в 6 ходов не более 9×8×7×6×5×4 (не более, потому, что в некоторых партиях после 5 хода наступает победа первого игрока и они не могут быть продолжены). Партий длиной в 7 ходов не более 9×8×7×6×5×4×3 (по тем же обстоятельствам, к которым добавляются партии выигранные вторым игроком на 6-м ходу). Партий длиной в 8 ходов не более 9! (часть партий не может быть продолжена после 5-го, 6-го или 7-го ходов). Партий длиной в 9 ходов не более 9! (часть партий не может быть продолжена после 5-го, 6-го, 7-го или 8-го ходов). Оценить сверху количество всевозможных партий можно, сложив все эти числа. Сумма равна 1172850 < 2^{21} = 2097152. Значит для записи номера партии достаточно 21 бита. Выберем такой способ нумерации, чтобы партия из условия была первой, тогда её запись будет такой: 00000000000000000001. Неважно какие номера у остальных партий, ведь в задаче требуется найти запись лишь для одной заданной.

Реальное количество законченных и незаконченных партий меньше найденной нами грубой оценки сверху 1172850. 1440 партий завершаются на пятом ходу, 5328 – на шестом, 47952 – на седьмом, 72576 – на восьмом, 81792 – на девятом. То есть количество партий длиной 6 ходов (завершённых и незавершенных) составляет 54720, длиной 7 ходов (завершённых и незавершенных) – 148176, длиной 8 ходов (завершённых и незавершенных) – 154368. Общее количество завершённых и незавершённых партий составляет 457786. Получается, что номер партии занимает 19 бит. В качестве записи заданной в условии партии можно указать любое число от 0 до 457785, записанное в двоичной системе в 19 битах.

При составлении задачи предполагалось, что грубой оценки сверху для решения задачи достаточно.

Критерий к задаче 3: Полный балл ставился за способ записи, обеспечивающий однозначность расшифровки, и использующий не более 32 бит для записи партии из условия задачи.

Задача 4. Счетные палочки. Решение

Идея решения заключается в том, чтобы сначала посимвольно считать набор палочек первого игрока, а затем набор палочек второго. При чтении в массиве N из пяти элементов-чисел нужно подсчитывать для первого игрока количество встретившихся палочек каждого типа (в N[i] должно оказаться количество палочек i-го типа, i = 1,5, массив N проинициализирован нулями). Аналогично в массиве M нужно подсчитать палочки второго игрока. Заметим что хранить целиком данные наборы палочек не следует, так только понапрасну расходуется память. Если язык позволяет, можно сразу подсчитать суммы очков. Иначе, следующим шагом будет “нормализация” массивов N и M. Мы как-бы обмениваем палочки младших типов на палочки старших типов, учитывая их номиналы – количества обозначаемых палочками очков. То есть, обмениваем по пять палочек первого типа на одну палочку второго типа до тех пор пока в N[1] и в M[1] не останется меньше пяти палочек. Аналогично палочки второго типа обмениваются на палочки третьего (по курсу пять к одной), палочки третьего типа обмениваются на палочки четвертого (по курсу десять к одной), палочки четвёртого типа обмениваются на палочки пятого (по курсу десять к одной). После “нормализации” мы можем сравнивать массивы N и M поэлементно, не вычисляя сумму очков. Сравнение начинается с пятых элементов массива, если они не равны, то победил тот игрок, у которого палочек пятого типа больше. Иначе сравниваем четвёртые элементы по тому же принципу и т. д.. Когда сравнение доходит до первых элементов, в случае их равенства выводится результат 0, иначе номер игрока, у которого больше палочек первого типа (следовательно, и очков).

Фрагменты программы на FreePascal:

```
program Task4 (input, output);
var     rate : array [1..5] of integer = (5, 5, 10, 10, 1); {массив для “конвертации” палочек}
        N, M : array [1..5] of longint;
        i: integer;
        res : byte;
        ch : char;
begin
        for i:=1 to 5 do begin
                N[i]:= 0;
                M[i]:= 0
        end; {for}
        {посимвольный ввод палочек первого игрока в массив N}
        while (not eoln) do begin
                read(ch);
```

```
            case ch of
            '1', '2', '3', '4', '5' : N[ord(ch) - ord('0')]:= N[ord(ch) - ord('0')] + 1
            end; {case}
        end; {while}
        readln;  {переход на следующую строку}
        {посимвольный ввод палочек второго игрока в массив M}
        while (not eoln) do begin
            read(ch);
            case ch of
            '1', '2', '3', '4', '5' : M[ord(ch) - ord('0')]:= M[ord(ch) - ord('0')] + 1
            end; {case}
        end; {while}
        for i:=1 to 4 do begin {"нормализация" массивов}
            N[i+1]:= N[i+1]+N[i] div rate[i];
            N[i]:= N[i] mod rate[i];
            M[i+1]:= M[i+1]+M[i] div rate[i];
            M[i]:= M[i] mod rate[i];
        end; {for}
        i:= 5;
        while (i > 1) and (N[i] = M[i]) do i:= i - 1; {поэлементное сравнение массивов}
        res:= 0;
        if (N[i] > M[i]) then res:= 1 else if (N[i] < M[i]) then res:= 2;
        write(res)
end.
```

Критерий к задаче 4: Полный балл ставился программу (текстовое описание алгоритма или блок-схему), решающую задачу, не содержащую логических и/или программных ошибок.

10-11 классы

Результаты технической проверки работ по задачам можно найти на странице участника http://ejudge.cs.msu.ru/new-client?contest_id=21 по логину и паролю, выданным на олимпиаде.

Задача A.

Задача в обоих вариантах оценивалась по набору из 13 тестов. Чем больше тестов пройдено программой, тем больший балл она получает за этой задание. Пример решения:

```
#include <iostream>
#include <sstream>
#include <vector>
#include <algorithm>

using namespace std;

int n;
std::vector<int> cost(10);
std::vector<int> sum;
vector<int> ans;
string s;

int main() {
    cin >> n;
    for (int i = 0; i < 10; ++i) {
        cin >> cost[i];
    }
    getline(cin, s);
    for (int i = 0; i < n; ++i) {
        getline(cin, s);
        stringstream scin(s);
        int tmp;
        int cs = 0;
        while (scin >> tmp) {
            cs += cost[tmp];
        }
        sum.push_back(cs);
    }

    int gmax = *min_element(sum.begin(), sum.end()); // or max_element()

    for (int i = 0; i < n; ++i) {
        if (sum[i] == gmax) {
            ans.push_back(i);
        }
    }

    for (int i = 0; i < (int)ans.size(); ++i) {
        cout << ans[i] + 1 << " ";
    }
    return 0;
}
```

Задача B.

Ответ в варианте 1: 542 2694833998866
Ответ в варианте 2: 639 633238162450

Для получения полного балла требовалось обоснование ответа. В случае его отсутствия за подсчет лишь одного из деревьев ставился неполный балл.

Задача C.

См. решение задачи 3 для 5-9 классов.

При проверке учитывалась корректность функций и длина получающегося кода. В зависимости от длины получается полный или частичный балл. Так же частично оценивались решения с в корректным кодированием и ошибками раскодирования. Решения, которые кодировали только один ход, считались неверными.

Задача D.

Пример одной из правильных программ.

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <cstdlib>
#include <cstring>
#include <cassert>

using namespace std;

const int MAXN = 1000;

const int SEEN = 1;
const int AVAILABLE = 2;
const int NOT_AVAILABLE = 4;
const int USED = 8;

int field[MAXN * 2 + 1][MAXN * 2 + 1];
const int DX[] = {-1, 0, 1, 0};
const int DY[] = {0, 1, 0, -1};
const char *DC = "LURD";

bool used[MAXN * 2 + 1][MAXN * 2 + 1];

int fly_cnt;
```

```
bool
check_pos(int x, int y, int op)
{
   return field[x][y] & op;
}

void
set_pos(int x, int y, int op)
{
   field[x][y] |= op;
}

void
analyze_cur_position(int x, int y)
{
   for (int dy = 1; dy >= -1; --dy) {
       for (int dx = -1; dx <= 1; ++dx) {
           int cur_x = x + dx, cur_y = y + dy;
           int cur_bit;
           set_pos(cur_x, cur_y, SEEN);
           cin >> cur_bit;
           if (cur_bit) {
               assert(!check_pos(cur_x, cur_y, NOT_AVAILABLE));
               set_pos(cur_x, cur_y, AVAILABLE);
           } else {
               assert(!check_pos(cur_x, cur_y, AVAILABLE));
               set_pos(cur_x, cur_y, NOT_AVAILABLE);
           }
       }
   }
}

void
land(int x, int y)
{
   if (fly_cnt > 0) {
       fly_cnt = 0;
       cout << 'P' << endl;
       analyze_cur_position(x, y);
   }
}

void
make_move(int dir, int x, int y)
{
   cout << DC[dir] << endl;
   analyze_cur_position(x, y);
   ++fly_cnt;
```

```
   fly_cnt %= 8;
}

int
get_reverse_dir(int cur_dir)
{
   return (cur_dir + 2) % 4;
}

void
dfs(int x, int y, int last_move_dir = -1)
{
   set_pos(x, y, USED);
   for (int dir = 0; dir < 4; ++dir) {
       int cur_x = x + DX[dir];
       int cur_y = y + DY[dir];
       assert(check_pos(cur_x, cur_y, SEEN));
       if (check_pos(cur_x, cur_y, AVAILABLE) && !check_pos(cur_x, cur_y, USED)) {
           make_move(dir, cur_x, cur_y);
           dfs(cur_x, cur_y, dir);
       }
   }
   if (last_move_dir != -1) {
       int back_dir = get_reverse_dir(last_move_dir);
       x += DX[back_dir], y += DY[back_dir];
       make_move(back_dir, x, y);
   }
}

int
get_dir(char char_dir)
{
   return find(DC, DC + 4, char_dir) - DC;
}

bool
fly_diagonal(int x, int y, const char *fly_pattern)
{
   for (int i = 0; i < 4; ++i) {
       int cur_dir = get_dir(fly_pattern[i]);
       x += DX[cur_dir];
       y += DY[cur_dir];
       make_move(cur_dir, x, y);
   }
   for (int dir = 0; dir < 4; ++dir) {
       int next_x = x + DX[dir], next_y = y + DY[dir];
       if (!used[next_x][next_y] && !check_pos(next_x, next_y, USED) && check_pos(next_x,
next_y, AVAILABLE)) {
           // second continent! YAHOO!!!
```

```
            make_move(dir, next_x, next_y);
            land(next_x, next_y);
            return true;
        }
    }
    for (int i = 3; i >= 0; --i) {
        int cur_dir = get_reverse_dir(get_dir(fly_pattern[i]));
        x += DX[cur_dir];
        y += DY[cur_dir];
        make_move(cur_dir, x, y);
    }
    return false;
}

bool
fly_straight(int x, int y, const int dir)
{
    for (int i = 0; i < 4; ++i) {
        x += DX[dir];
        y += DY[dir];
        make_move(dir, x, y);
        for (int ndir = 0; ndir < 4; ++ndir) {
            int next_x = x + DX[ndir], next_y = y + DY[ndir];
            if (!used[next_x][next_y] && !check_pos(next_x, next_y, USED) &&
check_pos(next_x, next_y, AVAILABLE)) {
                // second continent! YAHOO!!!
                make_move(ndir, next_x, next_y);
                land(next_x, next_y);
                return true;
            }
        }
    }
    for (int i = 0; i < 4; ++i) {
        int back_dir = get_reverse_dir(dir);
        x += DX[back_dir];
        y += DY[back_dir];
        make_move(back_dir, x, y);
    }
    return false;
}

bool
search_dfs(int x, int y, int last_move_dir = -1)
{
    used[x][y] = 1;
    for (int dir = 0; dir < 4; ++dir) {
        int next_x = x + DX[dir], next_y = y + DY[dir];
        if (check_pos(next_x, next_y, USED) && !used[next_x][next_y]) {
            make_move(dir, next_x, next_y);
```

```
            bool ret = search_dfs(next_x, next_y, dir);
            if (ret) {
                return true;
            }
        }
    }
    for (int dir = 0; dir < 4; ++dir) {
        land(x, y);
        if (fly_straight(x, y, dir)) {
            return true;
        }
    }
    const char *moves[] =
    {
        "LLDD",
        "LLUU",
        "RRDD",
        "RRUU"
    };
    for (int i = 0; i < 4; ++i) {
        land(x, y);
        if (fly_diagonal(x, y, moves[i])) {
            return true;
        }
    }
    if (last_move_dir != -1) {
        int back_dir = get_reverse_dir(last_move_dir);
        x += DX[back_dir], y += DY[back_dir];
        make_move(back_dir, x, y);
    }
    return false;
}

int
main()
{
    ios_base::sync_with_stdio(false);
    for (int i = 0; i < MAXN * 2 + 1; ++i) {
        for (int j = 0; j < MAXN * 2 + 1; ++j) {
            field[i][j] = 0;
        }
    }
    // make start point in the center of the field
    int root_x = MAXN, root_y = MAXN;
    fly_cnt = 0;
    // traverse all available cells - cells of the first continent
    analyze_cur_position(root_x, root_y);
    dfs(root_x, root_y);
    // land kvadrocopter if it in the air now
```

```
    land(root_x, root_y);
    bool res = search_dfs(root_x, root_y);
    assert(res);
    cout << 'W' << endl;
    return 0;
}
```

Задача E.

Данная функция возвращает -1 и числа от 2 до N, квадраты которых меньше 1024. Поэтому правильный ответ: 31. За ответ 32 и 33 ставились частичные баллы (не учтено, что функция не возвращает 0 и не возвращает 1). Для получения полного балла требовалось обосновать ответ.

Задача F.

Ответ: acbzefd.

Данная программа выводит следующую перестановку.

```
#include <iostream>
#include <algorithm>

int main() {
    std::string s;
    std::getline(std::cin, s);
    std::next_permutation(s.begin(), s.end());
    std::cout << s << std::endl;
}
```

Частичные баллы давались за неполное описание работы программы.