Задача 1. Робот-пылесос

Правильный ответ: $2 * w * h - 2 * w - 2 * h + 4$.

Робот-пылесос должен побывать во всех клетках склада, которых $w * h$. Дополнительно ему придётся ещё по разу побывать в клеточках, отмеченных звёздочкой. Их можно разбить на вертикальные «столбики», которых будет ровно $w - 2$ (без двух крайних, левой и правой сторон склада), а высота каждого столбика будет равна $h - 2$ (без последней клетки тупика, образованного полками, и клетки за ним).

Окончательно приходим к формуле $w * h + (w - 2) * (h - 2)$ или любой другой, получающейся после корректных алгебраических преобразований.

Задача 2. Маскарад

Ответ:

Захар принёс с собой грибы, он сделал маску кенгуру.

Белла принесла ягоды, она сделала маску орла.

Егор принёс собой фрукт, он сделал маску бегемота.

Харитон принёс с собой орехи, он сделал маску медведя.

Решение:

Исходя из первого утверждения, Харитон изготовил маску медведя.

Егор создал маску Бегемота, это мы видим по третьему утверждению.

Белла сделала маску орла. Согласно второму утверждению девочка изображает хищника; хищников только два – орёл и медведь, но маску медведя создал Харитон.

Захар сделал маску кенгуру.

Исходя из второго утверждения, можно сделать вывод, что Харитон принёс с собой орехи.

Исходя из четвёртого утверждения, можно сделать вывод, что Белла принесла с собой ягоды.

Егор принёс фрукты, так как, согласно третьему утверждению, Егор с собой не приносил ни грибов, ни ягод, а орехи уже принёс Харитон.

Захар принёс с собой грибы.

Задача 3. Разрезание строки

Наибольшей буквой в строке «СИРИУСОЛИМП» является буква «У» (она в этом слове одна), следующей буквой является «С» (их две).

Чтобы строка после разрезания и перестановки оказалась наибольшей в лексикографическом порядке, необходимо на первое место поставить букву «У». Значит, буква «У» должна быть началом одного куска, то есть разрез необходимо сделать перед буквой «У»: «СИРИ-УСОЛИМП». После перестановки получаем ответ на первую подзадачу: «УСОЛИМПСИРИ». Этот ответ приносит 25 баллов, следующий по лексикографическому порядку «СОЛИМПСИРИУ» – 15 баллов, следующий по лексикографическому порядку «РИУСОЛИМПСИ» – 5 баллов.

Удобно, что после буквы «У» сразу идёт буква «С», поэтому, вырезав их в паре, мы сможем улучшить результат вторым разрезанием: «СИРИ-УС-ОЛИМП». После перестановки второго куска в начало получаем ответ на вторую подзадачу: «УССИРИОЛИМП». Этот ответ приносит 25 баллов, следующий по лексикографическому порядку «УСОСИРИЛИМП» – 15 баллов, следующий по лексикографическому порядку «УСОЛСИРИИМП» – 5 баллов.

В первом куске предыдущего ответа за буквой «С» следует буква «И». В то же время третий кусок начинается с буквы «О». Выгоднее отрезать от первого куска маленькие буквы и вставить на их место третий кусок, а эту часть переместить в самый конец: «С-ИРИ-УС-ОЛИМП». После перестановок получаем ответ на третью подзадачу: «УССОЛИМПИРИ». Этот ответ приносит 25 баллов, следующий по лексикографическому порядку «УССИРОЛИМПИ» – 15 баллов, следующий по лексикографическому порядку «УССИРИПОЛИМ» – 5 баллов.

Длина исходного слова равна 11. Если переставить буквы в слове «СИРИУСОЛИМП» в неубывающем порядке, получим строку «УССРПОМЛИИИ». В этой строке, так же как в исходном слове рядом стоят пары букв «УС» и «ЛИ». Значит, при разрезании можно не разделять эти буквы, поэтому можно обойтись не десятью разрезами, а всего восьмью. Этот ответ приносит 25 баллов, девять разрезов – 15 баллов, ответ десять разрезов – 5 баллов.

Задача 4. Головоломка с простыми числами

Задачу можно решить двумя способами: при помощи программного перебора всех таблиц 3×3 либо при помощи математических рассуждений.

Для первого пути решения нужно написать программу на одном из языков программирования, состоящую в самом простом варианте из 9 циклов, перебирающих всевозможные таблицы с элементами от 0 до 10. Количество таких таблиц равно 11^9, что составляет 2 357 947 691 вариант. Внутри каждого варианта нужно найти искомые шесть сумм по строкам и столбцам, выбрать из них простые числа (все получающиеся простые числа не превосходят 30, их лучше заранее вручную просто внести в массив). Далее нужно найти сумму различных простых и сохранить максимальный вариант. Полный перебор всех вариантов такая программа может сделать в пределах одного часа, но один из оптимальных вариантов находится за две-три минуты. Например, такой:

$$\begin{matrix} 0 & 0 & 5 \\ 3 & 7 & 9 \\ 10 & 10 & 9. \end{matrix}$$

Но предпочтительнее второй вариант, использующий математические рассуждения. Так как сумма трех чисел от 0 до 10 не может превышать 30, то нас будут интересовать простые числа только в этих пределах. Попробуем взять шесть самых больших простых таких чисел: 29, 23, 19, 17, 13 и 11. Можно показать, что нельзя построить искомую таблицу, у которой шесть сумм по строкам и столбцам будут равны всем этим числам. Действительно, сумма по строкам равна сумме по столбцам, то есть некоторые три из этих чисел должны равняться в сумме оставшимся трём и эта сумма должна быть равна половине их общей суммы. Если взять число $(29 + 23 + 19 + 17 + 13 + 11)/2$, то получим число 56. Очевидно, что сумма трех нечётных простых чисел не может быть равной чётному числу. Попробуем уменьшить какое-нибудь из простых чисел. Можно попробовать заменить число 11 на число 7, но это снова приведёт тому же итогу, так как $(29+23+19+17+13+7)/2 = 54$. Следующие варианты — либо заменить число 13 на число 7, либо заменить число 11 на число 5. Оба они дадут одну и туже сумму 106, половина которой уже нечётна. Далее нужно попробовать разбить одно из этих множеств на две части так, что сумма внутри каждого была равна 53.

Рассмотрим вариант $29, 23, 19, 17, 11, 7$. Одна из половин должна содержать число 29, то есть оставшиеся две суммы должны вместе давать $53 - 29$, то есть 24. Легко заметить, что числа 17 и 7 как раз и дают такое значение. Тогда можно попробовать составить таблицу, у которой строки в суммах дают 29, 17 и 7, а столбцы – оставшиеся 23, 19 и 11. Имеется очень много вариантов построения такой таблицы, например, это может быть

$$\begin{matrix} 10 & 10 & 9 \\ 8 & 7 & 2 \\ 5 & 2 & 0. \end{matrix}$$

Если взять вариант $29, 23, 19, 17, 13, 5$, то для него также легко найти таблицу. Также ищем два из предложенных чисел, дающие в сумме 24, это 19 и 5. Строим таблицу, у которой суммы по строкам равны 29, 19 и 5, а по столбцам – 23, 17 и 13. Например, такую:

$$\begin{matrix} 9 & 10 & 10 \\ 3 & 6 & 10 \\ 1 & 1 & 3. \end{matrix}$$

Из этих рассуждений очевидно, что сумма 106 будет максимальной из возможных. Стоит отметить, что любая другая таблица, содержащая числа от 0 до 10 и дающая в суммах по строкам и столбцам это число, будет оцениваться в 100 баллов. Любые другие варианты, дающие в сумме по различным простым числам более чем 8, также получат пропорциональные полученной сумме баллы.

Задача 5. Всё могут короли!

Первая подзадача (полный перебор). При наиболее плотной «упаковке» короли будут располагаться в левых нижних клетках квадратов 2×2, которые замостят доску, начиная с левого нижнего угла. При этом сами короли будут располагаться исключительно на полях, обе координаты которых — нечётные числа. Переберём вложенным циклом все поля доски $n \times n$ и посчитаем количество подходящих.

```
n = int(input())
ans = 0
for i in range(1, n + 1):
    for j in range(1, n + 1):
        if i % 2 == 1 and j % 2 == 1:
            ans += 1
print(ans)
```

Данное решение имеет сложность $O(n^2)$.

Вторая подзадача (сокращение перебора). Заметим, что короли располагаются в квадратной сетке и что по вертикали будет столько же королей, сколько по горизонтали. Значит, можно просто возвести в квадрат это количество.

```
n = int(input())
ans = 0
for i in range(1, n + 1):
    if i % 2 == 1:
        ans += 1
ans = ans ** 2
print(ans)
```

Данное решение имеет сложность $O(n)$

Полное решение

Будем размещать королей, начиная с левого нижнего угла. Тогда их количество на нижнем крае доски будет равно $\lceil \frac{n}{2} \rceil$, где скобки означают округление вверх до целой части. Такое же количество королей окажется на левом крае доски, а значит, их общее число составит $\lceil \frac{n}{2} \rceil^2$.

```
n = int(input())
ans = ((n + 1) // 2) ** 2
print(ans)
```

Данное решение имеет сложность $O(1)$.

Задача 6. Натуральный ряд

Первая подзадача (полный перебор). Промоделируем описанную ситуацию: для каждого натурального числа установим, будет оно вычеркнуто или нет (последнее возможно, если число имеет ненулевой остаток при делении как на 2, так и на 3). Заведём счётчик подходящих чисел и будем его увеличивать, пока не наберём нужное количество.

```
n = int(input())
count = 0
number = 0
while count < n:
    number += 1
    if number % 2 > 0 and number % 3 > 0:
        count += 1
print(number)
```

Данное решение имеет сложность $O(n)$.

Полное решение: из каждых шести чисел, идущих подряд, останется два (имеющих остатки 1 и 5 при делении на 6), поэтому каждое увеличение n на 2 вызовет увеличение ответа на 6. Определим количество полных пар для числа $n-1$ и скорректируем ответ в зависимости от чётности самого n.

ans = (n - 1) // 2 * 6.

Если n нечётное, то к ответу добавим 1, иначе – 5.

```
n = int(input())
ans = (n - 1) // 2 * 6
```

```
if n % 2:
    ans += 1
else:
    ans += 5
print(ans)
```

Данное решение имеет сложность $O(n)$.

Задача 7. Городки

Первая подзадача: $n \leqslant 3$. Рассмотрим простые случаи.

При $n = 1$ есть только палочки единичной длины. Их длина и количество будет ответом, поскольку распилить такие палочки на две части нельзя (формально можно, но получатся части длиной 0 и 1).

При $n = 2$ есть палочки единичной длины и длины 2. Пусть у нас x палочек длины 1 и y длины 2. Попытаемся максимизировать количество палочек длины 1: их можно сделать $x + 2 \times y$ (добавляем распиленные пополам палочки длины 2). Попытаемся максимизировать количество палочек длины 2: их можно оставить только y. Первый результат лучше для любых неотрицательных значений. Поэтому ответом будет пара чисел $x + 2 \times y$ и 1.

При $n = 3$ рассуждения аналогичны. Пусть у нас x палочек длины 1, y длины 2 и z длины 3. Попытаемся максимизировать количество палочек длины 1 – это $x + 2 \times y + z$ (добавляем распиленные пополам палочки длины 2 и получившиеся при распиле палочек длины 3 кусочки длины 1). Попытаемся максимизировать количество палочек длины 2 – это $y + z$. Попытаемся максимизировать количество палочек длины 3: их можно оставить только z. Первый результат опять лучше для любых неотрицательных значений. Поэтому ответом будет пара чисел $x + 2 \times y + z$ и 1.

```
n = int(input())
if n == 1:
    x = int(input())
    print(x)
if n == 2:
    x = int(input())
    y = int(input())
    print(x + 2 * y)
if n == 3:
    x = int(input())
    y = int(input())
    z = int(input())
    print(x + 2 * y + z)
print(1)
```

Первая подзадача: $n \leqslant 1000$. Полный перебор. Создадим вспомогательный массив всех возможных размеров, для каждого размера переберём все возможные способы сделать конкретное значение палочек внутренним циклом. Заполнив данный массив, найдём в нём максимальный элемент.

```
n = int(input())
L = [0]
for i in range(n):
    L.append(int(input()))
M = [L[i] for i in range(n + 1)]
for i in range(1, n + 1):
    for j in range(i + 1, n + 1):
        if j % 2 == 1:
            if j // 2 == i:
                M[i] += L[j]
            if j // 2 + 1 == i:
```

```
                M[i] += L[j]
        elif j // 2 == i:
            M[i] += 2 * L[j]
best_ans = 1
best_count = M[1]
for i in range(2, n + 1):
    if M[i] > best_count:
        best_count = M[i]
        best_ans = i

print(best_count)
print(best_ans)
```

Полное решение: Сокращение перебора, поиск максимума за один проход по массиву.

Для каждого i у нас есть $L[i]$ исходных заготовок.

Если $2 \times i - 1 \leqslant n$, то есть возможность распилить все палочки длины $2 \times i - 1$ и получить дополнительно $L[2 \times i - 1]$ палочек длины i;

Если $2 \times i \leqslant n$, то есть возможность распилить все палочки длины $2 \times i$ и получить дополнительно $2 \times L[2 \times i - 1]$ палочек длины i;

Если $2 \times i + 1 \leqslant n$, то есть возможность распилить все палочки длины $2 \times i + 1$ и получить дополнительно $L[2 \times i + 1]$ палочек длины i.

Пройдём по массиву и для каждого i определим количество палочек данного размера, которые возможно получить из палочек исходной длины и палочек длин $2 \times i - 1$, $2 \times i$ и $2 \times i + 1$, выберем наибольшее из всех полученных значений.

```
n = int(input())
L = [0]

for i in range(n):
    L.append(int(input()))

best_ans = 1
best_count = L[1]
if n > 1:
    best_count += 2 * L[2]
if n > 2:
    best_count += L[3]
for i in range(2, n + 1):
    count_cur = L[i]
    if 2 * i - 1 <= n:
        count_cur += L[2 * i - 1]
    if 2 * i <= n:
        count_cur += 2 * L[2 * i]
    if 2 * i + 1 <= n:
        count_cur += L[2 * i + 1]
    if count_cur > best_count:
        best_count = count_cur
        best_ans = i

print(best_count)
print(best_ans)
```

Это же решение можно записать короче, если вместо проверки выхода за границу массива добавить в массив ещё $n + 1$ дополнительный элемент, соответствующий палочкам большей длины, присвоив им нулевые значения.

```
n = int(input())
a = [0] + [int(input()) for i in range(n)] + [0] * (n + 1)
best_ans = 1
best_count = a[1] + 2 * a[2] + a[3]
for i in range(2, n + 1):
    count = a[i] + a[2*i-1] + 2 * a[2*i] + a[2*i+1]
    if count > best_count:
        best_count = count
        best_ans = i
print(best_count)
print(best_ans)
```