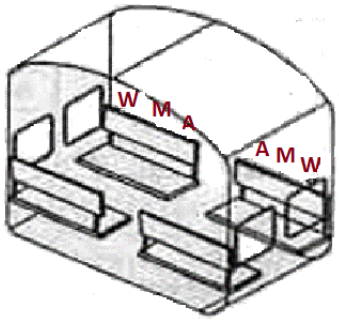
Горбунов Александр Евгеньевич

Балл: 134

Задача №1

Условие

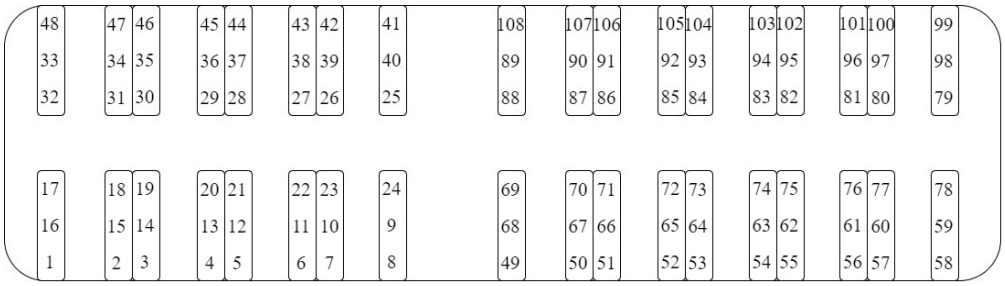
Миша и **Маша** очень любят путешествовать. В основном они путешествуют по железной дороге. Однажды они ехали в вагоне, где были только сидячие места, и заинтересовались расположением сидений в своем купе. Купе выглядело примерно так



Тип места обозначается следующим образом:

- Сиденье у окна: **W**
- Среднее место: **M**
- Место у прохода: **A**

Нумерация мест в вагоне приведена на следующем рисунке



Ребятам стало интересно узнать номер и тип сиденья, расположенного рядом с ними. В случае, когда сиденье расположено в середине, необходимо указать оба соседних номера.

Напишите программу, которая поможет ребятам решить эту задачу.

Входные данные

На вход подается одно целое число **N (**1 <= **N** <= 108**),** обозначающее номер места.

Выходные данные

Целое число, и большая латинская буква **W, M** или **A**, разделенные одним пробелом, которые будут соответствовать номеру и типу соседнего сиденья. Если таких сидений два, то данные по второму сидению выводится в той же строке через пробел (номера сидений в этом случае выводятся в порядке возрастания).

Примеры

Входные данные	Выходные данные
19	14 M
96	81 A 101 W

Исходный код

```
A = [
    [48, 47, 46, 45, 44, 43, 42, 41, 108, 107, 106, 105, 104, 103, 102, 101, 100, 99],
    [33, 34, 35, 36, 37, 38, 39, 40, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98],
    [32, 31, 30, 29, 28, 27, 26, 25, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79],
    [17, 18, 19, 20, 21, 22, 23, 24, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78],
    [16, 15, 14, 13, 12, 11, 10, 9, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59],
    [1, 2, 3, 4, 5, 6, 7, 8, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]
    ]


def char(i, j):
    if not i or i == 5:
        return 'W'
    if i == 1 or i == 4:
        return 'M'
    return 'A'


def end(i):
    return i == 5 or i == 2


def begin(i):
    return i == 0 or i == 3


def solve(i, j):
    ans = list()
    if end(i):
        ans.append([A[i - 1][j], char(i - 1, j)])
    elif begin(i):
        ans.append([A[i + 1][j], char(i + 1, j)])
    else:
        ans.append([A[i - 1][j], char(i - 1, j)])
        ans.append([A[i + 1][j], char(i + 1, j)])
    ans.sort()
    ans = ' '.join(' '.join(map(str, x)) for x in ans)
    print(ans)


n = int(input())
for i in range(len(A)):
    for j in range(len(A[i])):
        if A[i][j] == n:
            solve(i, j)
```

Задача №2

Условие

В кассе, продающей билеты по цене 500 р., нет возможности расплатиться картой, а в начале работы нет даже сдачи. Перед открытием у кассы собрались посетители, у каждого человека в очереди есть только одна купюра. У части – 500 р., у части – 1000 р. Сколькими способами можно выстроить посетителей в очередь так, чтобы к моменту обслуживания посетителя с купюрой в 1000 рублей у кассира всегда была сдача?

Напишите программу, которая решит эту задачу.

На вход программы подаются два целых неотрицательных числа:

n - количество человек с купюрой 500 р,

k - количество человек с купюрой 1000 р

Каждое число меньше 10.

Выведите одно целое число – сколькими способами можно выстроить очередь.

Пример

Входные данные	Выходные данные
2 1	4

Комментарий к примеру. Пусть есть посетители **A**, **B**, и **C**. У **A** и **B** по 500 рублёвой купюре, а у **C** – купюра в 1000 рублей. Тогда их можно выстроить так:

ABC – можно

ACB – можно

BAC – можно

BCA – можно

CAB – нельзя, т.к. в кассе к началу работы нет купюр на сдачу, а у первого посетителя купюра 1000р.

CBA – нельзя, т.к. в кассе к началу работы нет купюр на сдачу, а у первого посетителя купюра 1000р.

Есть только 4 возможных варианта.

Исходный код

```
from math import factorial


def func(n, k, sd):
    if (not n and not k or k < 0 or n < 0 or sd < 0):
        return 0
    if (n == 1 and not k):
        return 1
    if (k == 1 and not n):
        return bool(sd)
    return func(n - 1, k, sd + 1) + func(n, k - 1, sd - 1)


n, k = map(int, input().split()) # 500_1000
print(func(n, k, 0) * factorial(n) * factorial(k))
```

Задача №3

Условие

Исполнитель получает на вход натуральное число X (не превышающее 10^6). По этому числу, точнее по его представлению в шестеричной системе счисления строится новое число Y по следующим правилам.

В шестеричном представлении числа X предпоследняя цифра увеличивается на 1 (гарантируется, что в шестеричном представлении X числа больше 2-х цифр). Например, $749_{10} = 32\mathbf{4}5_6$ -> $32\mathbf{5}5_6 = 755_{10}$.

Если предпоследняя цифра 5, тогда предпоследняя цифра становится 0, а последняя изменяется по следующему принципу: четная увеличивается н 1, а нечетная уменьшается на 1. Например, последняя цифра нечетная $751_{10} = 32\mathbf{5}\boldsymbol{1}_6$ -> $32\mathbf{0}\boldsymbol{0}_6 = 720_{10}$, последняя цифра четная $752_{10} = 32\mathbf{5}\boldsymbol{2}_6$ -> $32\mathbf{0}\boldsymbol{3}_6 = 723_{10}$.

Введем понятие расстояния

Oh = | Исходное_число – Полученное_число |

Напишите программу, которая будет считать наибольшее расстояние Oh для чисел из заданного интервала [A,B] и наибольшее исходное число, для которого оно было вычислено.

На вход программы подаётся два целых числа A и B ($10 \le A \le B \le 1\,000\,000$), записанных через пробел.

Программа должна вывести два числа наибольшее расстояние Oh и через пробел исходное число, для которого оно было посчитано.

Входные данные	Вывод	Примечание
748 752	31 751	|748 - 754| = 6 ($32\mathbf{4}4_6 - 32\mathbf{5}4_6$) |749 - 755| = 6 ($32\mathbf{4}5_6 - 32\mathbf{5}5_6$) |750 - 721| = 29 ($32\mathbf{50}_6 - 32\mathbf{01}_6$) |751 - 720| = 31 ($32\mathbf{51}_6 - 32\mathbf{00}_6$) |752 - 723| = 29 ($32\mathbf{52}_6 - 32\mathbf{03}_6$)
747 749	6 749	|747 - 753| = 6 ($32\mathbf{4}3_6 - 32\mathbf{5}3_6$) |748 - 754| = 6 ($32\mathbf{4}4_6 - 32\mathbf{5}4_6$) |749 - 755| = 6 ($32\mathbf{4}5_6 - 32\mathbf{5}5_6$)

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
#define fixstd() ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0)
#define endl "\n"
#define pb push_back
#define sz(arr) int(arr.size())
#define all(arr) arr.begin(), arr.end()
#define print(arr) {for (auto el: arr) cout << el << " "; cout << endl;}
using namespace std;

vector<int> six(int n) {
 vector<int> ans;
 for (; n; n /= 6)
  ans.pb(n % 6);
 reverse(all(ans));
 return ans;
}

vector<int> po;

void countPo(int n=9) {
 po.resize(n + 1);
 po[0] = 1;
 for (int i = 1; i <= n; i++)
  po[i] = po[i - 1] * 6;
}

int back(vector<int> &x) {
 reverse(all(x));
 int n = sz(x);
 int ans = 0;
 for (int i = 0; i < n; i++)
  ans += x[i] * po[i];
 return ans;
}

int func(int n) {
 auto x = six(n);
 int s = sz(x);
 //print(x);
 if (x[s - 2] != 5) x[s - 2]++;
 else {
  x[s - 2] = 0;
  x[s - 1] += (x[s - 1] & 1 ? -1 : 1);
 }
 //print(x);
 //cout << endl;
 int m = back(x);
 return abs(n - m);
}

int main() {
 fixstd(); countPo();
 int a, b; cin >> a >> b;
 int ans = -1, num;
 for (int i = a; i <= b; i++) {
  int tmp = func(i);
```

```
  if (tmp >= ans) {
   ans = tmp;
   num = i;
  }
 }
 cout << ans << " " << num;
 //system("pause");
}
```

Задача №4

Условие

Зонд передает данные с орбиты Юпитера во время сильной магнитной бури. Информация передается по каналу связи в виде пакетов. Каждый пакет представляет собой целое положительное число в двоичной системе счисления. Для обнаружения помех последний разряд в пакете подбирают таким образом, чтобы количество единиц в разрядах пакета было четным. В каждом пакете на практике никогда не искажается больше одного разряда.

Напишите программу, которая по распечатке пакетов, записанных в десятичной системе счисления, найдет самое большое значение, прошедшее без искажений. Известно, что как минимум один пакет прошел без искажений.

Формат ввода

В строке вводится сначала целое число ***n*** – количество пакетов (***n ≤ 1000***), затем ***n*** натуральных чисел, все числа отделены друг от друга одним или несколькими пробелами.

Формат вывода

Вывести одно целое число – самое большое значение, прошедшее без искажений.

Пример

Входные данные	Выходные данные
4 1025 496 882 1056	1056

Исходный код

```
def func(n):
    b = list(map(int, bin(n)[2:]))
    print(b)
    return not (b.count(1) & 1)


arr = list(map(int, input().split()))
n = arr.pop(0)
ans = max(filter(func, arr))
print(ans)
```

Задача №5

Условие

Вася придумывает пароль для каждой новой учетной записи, которую он заводит на каком-то из своих устройств. Время от времени он изменяет пароль, дописывая к нему новые символы. Предыдущий пароль никогда не будет началом пароля для новой учетной записи Васи. После того, как Вася заводит новую учетную запись, он перестает менять пароль на старой.

Зная все Васины пароли в хронологическом порядке, напишите программу, которая найдет, какое наибольшее количество раз Вася менял пароль для одной учетной записи.

Формат ввода

В первой строке вводится сначала целое число ***n*** – количество слов (***n ≤ 1000***), затем в ***n*** следующих строках записано по слову. Слова состоят только из строчных латинских букв.

Формат вывода

Вывести одно целое число – какое наибольшее количество раз Вася менял пароль для одной учетной записи.

Примеры

Входные данные	Выходные данные
3 abd abdc bvd	1
4 bcd bcdf bcdfe abc	2

Исходный код

```
n = int(input())
pref = input()
ans = cnt = 0
for _ in range(n - 1):
    s = input()
    #print(s.find(pref))
    if not s.find(pref):
        cnt += 1
    else:
        ans = max(ans, cnt)
        cnt = 0
    pref = s
ans = max(ans, cnt)
print(ans)
```

Задача №6

Условие

Миша ведет дневник, отслеживая, сколько шагов в день он проходит пешком. Каждый день он записывает количество шагов. Требуется определить, в какой день во время периода наблюдений он прошел наибольшую дистанцию и прошел ли он столько, сколько планировал, за все время.

На вход программе в первой строке подаются натуральное число N – количество дней, в течение которых велось наблюдение ($N \leq 20$) и натуральное число X – запланированное количество шагов.

Далее в N строках подается на вход по целому положительному числу – x_i: количество шагов, пройденных в i-й день.

Вывести два целых числа – в первой строке вывести номер дня, в который Миша прошел больше всего шагов, во второй строке вывести 1, если он прошел не менее запланированного числа шагов и 0 в обратном случае. Если Вася прошел одинаковое количество шагов в несколько разных дней, вывести наименьший номер дня.

Пример

Ввод	Вывод

3 10000 5100 6500 4200	2 1

Исходный код

```
n, x = map(int, input().split())
arr = [int(input()) for _ in range(n)]
maind = arr.index(max(arr))
print(maind + 1)
print(int(sum(arr) >= x))
```

Задача №7

Условие

Исследовательский аппарат на поверхности Марса может выполнять команды «Фотографирование», «Пробное бурение», «Взятие образцов грунта», «Анализ атмосферы». Из-за конструктивных особенностей на аппарат наложен ряд ограничений. Нельзя выполнять команду «Анализ атмосферы» после команды «Пробное бурение». Нельзя выполнить команду «Фотографирование» после команды «Взятие образцов грунта». Команду «Взятие образцов грунта» можно выполнять только следующей после команды «Пробное бурение». Никакую команду, кроме команды «Взятие образцов грунта», нельзя выполнить подряд дважды.

Напишите программу, которая определит, сколько существует выполнимых последовательностей команд длиной n, если до начала выполнения программы аппарат выполнил команду «Пробное бурение».

На вход программе подается натуральное число n ($n \le 15$) – количество команд.

Вывести целое число – количество выполнимых последовательностей команд длиной n.

Пример

Ввод	Вывод
2	5

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
#define fixstd() ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0)
#define endl "\n"
#define pb push_back
#define sz(arr) int(arr.size())
#define all(arr) arr.begin(), arr.end()
#define print(arr) {for (auto el: arr) cout << el << " "; cout << endl;}
using namespace std;

int func(int x, int n) {
    if (!n) return 1;
    if (x == 0) // photographyrovanie
        return func(1, n - 1) + func(3, n - 1);
    else if (x == 1) // probnoe burenie
        return func(0, n - 1) + func(2, n - 1);
 else if (x == 2) // vzatie grunta
        return func(1, n - 1) + func(2, n - 1) + func(3, n - 1);
 else if (x == 3) // analiz atmosphery
        return func(0, n - 1) + func(1, n - 1);
}

int main() {
 fixstd();
 int n; cin >> n;
 cout << func(1, n) << endl;
 //system("pause");
}
```