Задача A. Ферзи

Задачу можно решить конструктивно, но также возможно решение метод отжига. Будем кодировать состояние так же перестановкой чисел от 0 до $(n-1)$: ферзь номер i будет стоять на пересечении i-той строки и p_i -того столбца.

Такое представление кодирует не все возможные расстановки, но это даже хорошо: точно не учтутся те расстановки, где ферзи бьют друг друга по вертикали или горизонтали.

Выберем функцию $f(p)$, равную числу успешно расставленных ферзей.

```
def f(placement, current_ans, x1, x2, d1, d2):
    y1, y2 = placement[x1] - 1, placement[x2] - 1
    d1, d2 = d1[:], d2[:]

    d1[x1-y1+n-1], d2[x1+y1] = d1[x1-y1+n-1]-1, d2[x1+y1]-1
    for d, i in zip([d1, d2], [x1-y1+n-1, x1+y1]):
        if d[i] == 1:
            current_ans -= 2
        elif d[i] > 0:
            current_ans -= 1

    d1[x2-y2+n-1], d2[x2+y2] = d1[x2-y2+n-1]-1, d2[x2+y2]-1
    for d, i in zip([d1, d2], [x2-y2+n-1, x2+y2]):
        if d[i] == 1:
            current_ans -= 2
        elif d[i] > 0:
            current_ans -= 1

    d1[x1-y2+n-1], d2[x1+y2] = d1[x1-y2+n-1]+1, d2[x1+y2]+1
    for d, i in zip([d1, d2], [x1-y2+n-1, x1+y2]):
        if d[i] == 2:
            current_ans += 2
        elif d[i] > 2:
            current_ans += 1

    d1[x2-y1+n-1], d2[x2+y1] = d1[x2-y1+n-1]+1, d2[x2+y1]+1
    for d, i in zip([d1, d2], [x2-y1+n-1, x2+y1]):
        if d[i] == 2:
            current_ans += 2
        elif d[i] > 2:
            current_ans += 1

    return current_ans, d1, d2
```

Теперь используя эту функцию начинаем отжиг, постепенно уменьшая температуру. В итоге с высокой вероятностью находим правильную расстановку.

```
def change(placement):
    ans = placement[:]
    ind = randint(0, len(ans)-1)
    ind2 = randint(0, len(ans)-1)

    if ind == ind2:
        return ans, f_theta, diagonals1, diagonals2

    f_new, d1, d2 = f(ans, f_theta, ind, ind2, diagonals1, diagonals2)

    ans[ind], ans[ind2] = ans[ind2], ans[ind]
```

```
    return ans, f_new, d1, d2

n = int(input())

theta = [i for i in range(1, n+1)]

diagonals1 = [0] * (2*n-1)
diagonals1[n-1] = n

diagonals2 = [0 if i%2==1 else 1 for i in range(2*n-1)]
f_theta = n

T = 100
while True:
    if f_theta == 0:
        print(*theta)
        break

    theta_new, f_theta_new, d1, d2 = change(theta)
    if f_theta_new <= f_theta or exp((f_theta - f_theta_new) / T) > random():
        theta = theta_new
        f_theta = f_theta_new
        diagonals1, diagonals2 = d1, d2
    T *= 0.99
```

Задача B. ТСП

Существует множество подходов к решению задачи. Не исключено, что подходящий ответ можно получить и простым перебором перестановок. Авторское решение предполагает использование отжига. За функцию $f(p)$ примем сумму всех расстояний в нашем оптимальном маршруте и будет пытаться ее оптимизировать.

```
def f(p, g):
    m = len(p)
    dist = 0
    for i in range(1, m):
        dist += g[p[i - 1]][p[i]]
    return dist
```

Менять состояние будем простой заменой мест элементов в перестановке. В итоге получаем решение, которое за какое-то адекватное время найдет оптимальное решение.

```
permutation = [0] * 312
while (last_answer > target):
    gen_permutation(312, permutation)
    t = 100
    k = 10 ** 6 * 2
    kk = 0
    cur_ans = ans = f(permutation, g)
    while kk <= k:
        i = random.randint(0, 71721) % 312
        j = random.randint(0, 2434324) % 312
        permutation[i], permutation[j] = permutation[j], permutation[i]
        cur_ans = f(permutation, g)
        if cur_ans <= ans or random.randint(0, 1) < math.exp((ans - cur_ans) / t):
            ans = cur_ans
```

```
        else:
            permutation[i], permutation[j] = permutation[j], permutation[i]
        t *= 0.99
        kk += 1
    last_answer = ans
print(' '.join(permutation))
```

Задача C. Хорошие раскраски

Заметим, что найдя раскраску 10 на 10, мы получим ответы и для всех перестановок меньшего размера. Одно из возможных решений - случайная раскраска поля, с последующей проверкой его на корректность. Запускаем перебор локально и находим нужную нам раскраску. Время сильно зависит от оптимизаций, которые используются в переборе. В целом на нахождение ответа не должно уйти больше получаса.

```
if n == 10 and m == 10:
   print('''2 1 3 3 1 3 2 1 2 1
1 2 1 3 2 1 3 3 2 1
2 1 3 2 3 1 3 2 1 3
1 3 2 1 2 3 2 1 1 3
3 3 3 2 2 2 1 3 1 1
1 1 2 2 3 3 1 3 2 2
2 3 1 3 3 2 1 1 3 2
2 2 2 1 1 1 1 3 3 3
3 1 1 1 2 3 3 2 3 2
3 2 1 3 1 2 2 2 1 3''')
```

Задача D. Бонус с блэкджеком

Задача сводится к написанию стратегии для игры в блэкджек. Сложность представляет скорее реализация, чем проработка стратегии. Ставить деньги случайно не получится. Нужно проработать логику и определять ситуации, когда рука сильная и можно ставить ставку и наоборот.

```
def is_soft(v):
       for x in v):
           if x == 11:
               return 1
       return 0
def play(hand)
   if (is_soft(hand)):
       if (score(hand) <= 17)
           hit()
           continue
       if (score(hand) >= 18):
           brk()
           break
   if (score(hand) >= 17):
       brk()
       break
   if (score(hand) < 11):
       hit()
       continue
   if (score(hand) == 11):
       dbl(bet)
       continue
...
```