Sorting the Shifts

Let us consider a device whose task is sorting a sequence of **n** numbers in increasing order. At each step of operation, the device performs the specified *conditional exchange* operation. A conditional exchange operation between positions **i** and **j** (1 <= **i** < **j** <= **n**) modifies the number sequence by swapping the value at the **i**-th position with the value at the **j**-th position if the former is initially larger than the latter. So, applying the conditional exchange operation to a sequence:

$\mathbf{a}_1, ..., \mathbf{a}_\mathbf{i}, ..., \mathbf{a}_\mathbf{j}, ..., \mathbf{a}_\mathbf{n}$

results in the sequence:

$\mathbf{a}_1, ..., \min\{\mathbf{a}_\mathbf{i},\mathbf{a}_\mathbf{j}\}, ..., \max\{\mathbf{a}_\mathbf{i},\mathbf{a}_\mathbf{j}\}, ..., \mathbf{a}_\mathbf{n}$

with all positions except for the **i**-th and the **j**-th unchanged.

It is easy to see that there exist series of conditional exchange operations which will sort any input sequence in increasing order. For example, for **n**=4, the following series of successive conditional exchange operations (specified through indices **i** and **j**), performed one by one, is a realisation of the well-known bubble-sort algorithm:

1 2
2 3
3 4
1 2
2 3
1 2

The task becomes a bit more entertaining when the device does not have to be able to sort all input sequences, but only some sequences of a very special form. In this problem we are interested in sorting sequences of integers which can be written in the form:

$(2^\mathbf{k}) \bmod \mathbf{n}, (2^\mathbf{k}+1) \bmod \mathbf{n}, (2^\mathbf{k}+2) \bmod \mathbf{n}, ..., (2^\mathbf{k}+n-1) \bmod \mathbf{n}$

for some value of integer **k** >=0, where "*a* mod *b*" denotes the remainder in the division of *a* by *b*.

So, for example, for **n**=4 the device must be able to sort sequences of the form:

1, 2, 3, 0 (**k**=0)
2, 3, 0, 1 (**k**=1)
0, 1, 2, 3 (**k**=2)
0, 1, 2, 3 (**k**=3)
...

while for **n**=5, sequences of the form:

1, 2, 3, 4, 0 (**k**=0)
2, 3, 4, 0, 1 (**k**=1)
4, 0, 1, 2, 3 (**k**=2)
3, 4, 0, 1, 2 (**k**=3)
...

Your task is to design appropriate sorting devices for chosen small values of **n**, trying to use as few conditional exchange operations as possible.

Input

There is no input data for this problem.

Output

Output your designs of the specified sorting devices for any values of **n** you like from the range from 3 to 102 inclusive (so, you can output at most 100 devices). The designs should be written one after another.

The description of each device is as follows. In the first line output integer **n** ($3 \leq \mathbf{n} \leq 102$). In the second line output integer **m** ($1 \leq \mathbf{m} \leq \mathbf{n}^2$), the number of conditional exchange operations which appear in your device. In each the next **m** lines, output exactly one space separated pair of indices **i** and **j** ($1 \leq \mathbf{i} < \mathbf{j} \leq \mathbf{n}$), describing the conditional exchange operation.

For all sorted sequences, the operations will be performed in the order in which you provide them. The device must sort all of the considered sequences correctly, and must fulfill the stated constraints, otherwise your program will be judged incorrect.

Points

For a successfully designed device for sorting permitted sequences of length **n** with **m** conditional exchange operations you will receive **n**/**m** points. The total score of your program is the sum of scores obtained for individual devices.

The number of points given in the ranking is scaled so that it is equal to 10 for the registered contestant whose solution has the highest score, and proportionally less for all solutions with lower scores.

Example 1

Output:
3
3
1 2
2 3
1 2
4
6
1 2
2 3
3 4
1 2
2 3
1 2

Score:
5/3 pts.

The devices in the above example simply perform a bubble-sort for **n**=3 and **n**=4, receiving 1 point and 2/3 pts. for each of these devices, respectively. Clearly, this is not the best way of solving the problem, but it is a start.

Example 2

Output:
4
5
2 3
3 4
1 2
2 3
1 2

Score:
4/5 pts.

If you look carefully at the allowed sequences for **n**=4, you will notice that in Example 1 the first conditional exchange (1 2) never results in the swapping of values. So, we can safely leave it out, improving the score for this device from 2/3 to 4/5.

Notes

- The source code length limit for this problem is 8kB.
- For the last week of the series, submissions will be visible to the submitting contestant, only.