Problem A. Test

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Consider all binary string of length n. Chiaki would like to arrange them in the decreasing order of the number of 1s in the string. In addition, the number of difference bits of two adjacent strings should not exceed 2.

Input

The first line contains an integer n ($1 \leq n \leq 16$) denoting the length of the binary strings.

Output

Output 2^n lines. Each line contains a binary string of length n. They should be ordered according to the rule above.

Examples

standard input	standard output
1	1 0
2	11 10 01 00

Problem B. Wheel

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Chiaki has a wheel graph with $n+1$ vertices, which is formed by connecting a single vertex to all vertices of a cycle of length n.

Each vertex will be colored into white or black with equal probability. Chiaki would like to know expectation of the size of the largest connected component with same color.

For example, consider the wheel graph with 4 vertices. Then with probability $\frac{1}{8}$ all vertices will be the same color. In this case, the maximum size of is 4. With probability $\frac{3}{8}$, two vertices will be black, and the other two will be white. In this case, the maximum size of is 2. Finally, with probability $\frac{1}{2}$, one vertex will be white (black), the other three will be black (white). In this case, the maximum size will be 3. The mathematical expectation of the answer in this case is $4 \times \frac{1}{8} + 2 \times \frac{3}{8} + 3 \times \frac{1}{2} = 2.75$.

Input

There are multiple test cases. The first line of input contains an integer T denoting the number of test cases. For each test case:

The first line contains an integer n ($3 \leq n \leq 500$).

It is guaranteed that the sum of n does not exceed 1000.

Output

For each test case, output a real number denoting the answer. The answer is correct if it differs from the correct one by no more than 10^{-9}.

Example

standard input	standard output
2 3 4	2.75 3.4375

Problem C. Equality

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 megabytes

Chiaki has n integers $a_1, a_2, \ldots, a_n$. She would like to find a non-empty subset S of $\{1, 2, \ldots, n\}$ such that

$$\sum_{v \in S} v = \sum_{v \in S} a_v.$$

Input

There are multiple test cases. The first line of input contains an integer T denoting the number of test cases. For each test case:

The first line contains an integer n ($1 \leq n \leq 10^6$) – the number of integers. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$).

It is guaranteed that the sum of n does not exceed 10^6.

Output

For each test case, output the size of the subset in the first line. Then in the second line, output the numbers in the subset. If you cannot find such subset, just output -1 in a single line.

Example

standard input	standard output
1 5 3 4 1 2 2	2 1 3

Problem D. Quick Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Consider the following quick sort functions:

```
function qqsort(A: array of int): array of int
    if isSorted(A):
        return A
    middle = pick(A)
    aLess = elements of A less then middle
    aGreater = elements of A greater then middle
    return qqsort(aLess) + [middle] + qqsort(aGreater)
```

To sort an array, the following is done: first, it is checked whether it is already sorted. If it is, no further action is required. Otherwise, some element of the array is selected, the rest are divided into those that are smaller than it and those that are larger, after which the sorting for these new arrays is started. In new arrays, the elements are in the same order as they were in the source array.

The *isSorted(a)* function checks if the array is already sorted. The *pick* function returns some element from the array, which is passed to it as an argument. It can return any of the elements of the array.

It is known that the speed of sorting depends on how well the dividing element is chosen. Accordingly, depending on the values returned by the *pick* function, the number of actions that can be performed while the algorithm is running can change.

Chiaki has an array of n different integers from 1 to n. She would like to know the minimum number of calls to function *pick* in order to sort the array, if you can determine the return value of each call to function *pick*.

For example, if you want to sort the array $A = [1, 2, 6, 7, 5, 3, 4, 8, 9]$, if the first call to pick returns 5, then the array $aLess = [1, 2, 3, 4]$, and the array $aGreater = [6, 7, 8, 9]$. Accordingly, recursive calls to *qqsort* will immediately exit, because in them *isSorted* calls will return true and the array will be sorted. This is optimal, since the pick function was called only once.

Input

There are multiple test cases. The first line of input contains an integer T denoting the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^6$) – the number of integers. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$, for $i \ne j$, $a_i \ne a_j$).

It is guaranteed that the sum of n does not exceed 10^6.

Output

For each test case, output an integer denoting the answer.

Example

standard input	standard output
3 9 1 2 6 7 5 3 4 8 9 4 4 3 2 1 2 1 2	1 2 0

Problem E. Split

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Consider the set of integers from 1 to $3n$. Chiaki would like to distribute these numbers into three arrays a, b and c of length n such that for any i from 1 to n the following is true: $a_i + b_i = c_i$.

Input

The first line contains an integer n ($1 \leq n \leq 23$).

Output

If you cannot find a solution, output -1 in a single line. Otherwise, output three lines and each should contain n integers separated by spaces. The first line should contain elements of array a, in the second – elements of array b, in the third – elements of array c. Each number from 1 to $3n$ must be displayed exactly once.

Example

standard input	standard output
1	1 2 3