

At the IT Campus "NEIMARK", there are several top-secret rooms where problems for major programming competitions are developed. To enter one of these rooms, you must unlock a circular lock by selecting the correct code. This code is updated every day.

Today's code is a permutation* of the numbers from 1 to n , with the property that in every cyclic shift† of it, there is exactly one fixed point. That is, in every cyclic shift, there exists exactly one element whose value is equal to its position in the permutation.

Output any valid permutation that satisfies this condition. Keep in mind that a valid permutation might not exist, then output -1 .

*A permutation is defined as a sequence of length n consisting of integers from 1 to n , where each number appears exactly once. For example, $(2\ 1\ 3)$, (1) , $(4\ 3\ 1\ 2)$ are permutations; $(1\ 2\ 2)$, (3) , $(1\ 3\ 2\ 5)$ are not.

†A cyclic shift of an array is obtained by moving the last element to the beginning of the array. A permutation of length n has exactly n cyclic shifts.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 500$). The description of the test cases follows.

A single line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the desired permutation. If multiple solutions exist, output any one of them. If no suitable permutations exist, output -1 .

at the end what this problem transform to ?
tutorial

Let after k cyclic shifts the element p_i be at position $(i+k) \bmod n$. For this point to be fixed, it is necessary that $(i+k) \bmod n = p_i$. Let assume that permutation p is starting from 0, $[0, 1, 2, \dots, n-1]$.

From this we get that $k = p_i - i \pmod n$. For any cyclic shift to be a fixed point — it must be possible to obtain any k from 0 to $n-1$. Let us sum both sides for all possible values of $k = p_i - i \pmod n$.

We get $\sum_{i=0}^{n-1} k = 0$ and $\sum_{i=0}^{n-1} k = \sum_{i=0}^{n-1} (p_i - i) = \sum_{i=0}^{n-1} p_i - \sum_{i=0}^{n-1} i = 0$.

In order to construct a permutation, these sums must be equal modulo n , i.e. $n \cdot (n-1)/2 \equiv 0 \pmod n$. For even n — this is impossible and the answer is -1 . And for odd n one of the options — $p = [n-1, n-2, \dots, 1]$. $O(n)$.

give full mathematical reasoning and logic of this tutorial

give all proof and code



Thought for 1m 31s >

The heart of the tutorial is this:

The problem is **equivalent** to finding a permutation p such that all values $p_i - i \pmod n$ are **distinct** and cover all residues $0, 1, \dots, n-1$. Such permutations exist **iff** n is odd, and one simple example is the reversed permutation.

Let's go step by step, fill in all the gaps, and then I'll give code (Python + C++).

1. Setting up the math (0-based indexing)

The original permutation is on $\{1, \dots, n\}$. It's much cleaner to switch to **0-based** indices and values:

- Positions: $i = 0, 1, \dots, n-1$.
- Values: also $p_i \in \{0, 1, \dots, n-1\}$, forming a permutation.

A **cyclic shift by k** means: each element moves from position i to position $(i + k) \bmod n$.

A **fixed point** in the shifted permutation means:
after shifting by k , some element equals its position.

Suppose the element originally at position i (value p_i) is a fixed point after shift k .

- After shift by k , it is at position $(i + k) \bmod n$.
- To be a fixed point:

$$(i + k) \equiv p_i \pmod{n}.$$

Rearrange:

$$k \equiv p_i - i \pmod{n}.$$

Define

$$k_i := (p_i - i) \bmod n.$$

So: **position i** contributes a fixed point in the shift **number k_i** .

2. "Exactly one fixed point per shift" \Rightarrow a bijection

The problem says:

For each cyclic shift $k = 0, 1, \dots, n - 1$, there is **exactly one** fixed point.

In terms of our $k_i = p_i - i \pmod{n}$, this means:

- For each shift value k (from 0 to $n - 1$), there is **exactly one index i** such that

$$k \equiv k_i \equiv p_i - i \pmod{n}.$$

So the mapping

$$i \mapsto k_i = (p_i - i) \bmod n$$

must be a **bijection** from $\{0, \dots, n - 1\}$ to $\{0, \dots, n - 1\}$.

Equivalently, the multiset $\{k_i\}$ must be exactly $\{0, 1, \dots, n - 1\}$.



Problem transformed:

Find a permutation p such that the numbers

$\{(p_i - i) \bmod n : i = 0, \dots, n - 1\}$
are all distinct and equal to the full set $\{0, \dots, n - 1\}$.

This is what the editorial is getting at when they say “for any cyclic shift to be a fixed point — it must be possible to obtain any k from 0 to $n - 1$ ”.

3. Summing the k 's: necessary modular condition

Since $\{k_i\}$ is a permutation of $\{0, \dots, n - 1\}$, we have:

$$\sum_{k=0}^{n-1} k = \sum_{i=0}^{n-1} k_i \pmod{n}.$$

Left-hand side

Sum of the first n integers:

$$\sum_{k=0}^{n-1} k = \frac{n(n-1)}{2}.$$

Right-hand side

$$\sum_{i=0}^{n-1} k_i = \sum_{i=0}^{n-1} (p_i - i) \pmod{n} = \left(\sum_{i=0}^{n-1} p_i \right) - \left(\sum_{i=0}^{n-1} i \right) \pmod{n}.$$

But p is a permutation of $\{0, \dots, n - 1\}$, so:

$$\sum_{i=0}^{n-1} p_i = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}.$$

Therefore

$$\sum_{i=0}^{n-1} (p_i - i) = \frac{n(n-1)}{2} - \frac{n(n-1)}{2} = 0.$$

So the equality of sums becomes:

$$\frac{n(n-1)}{2} \equiv 0 \pmod{n}.$$

This is **necessary** for any solution to exist.

The tutorial summarizes this as:

"we get $\sum k = \frac{n(n-1)}{2}$ and $\sum (p_i - i) = 0$. For a permutation to exist, these must be equal modulo n , i.e. $\frac{n(n-1)}{2} \equiv 0 \pmod{n}$."

4. When does $\frac{n(n-1)}{2} \equiv 0 \pmod{n}$ hold?

We need:

$$n \mid \frac{n(n-1)}{2}.$$

This is the same as:

$$\frac{n(n-1)}{2} = n \cdot \frac{n-1}{2},$$

so this is an integer only if $\frac{n-1}{2}$ is an integer (which is always true) **and** the divisibility condition is trivial:

- If n is **odd**:

$n - 1$ is even, so $\frac{n-1}{2}$ is an integer. Then

$$\frac{n(n-1)}{2} = \left(\frac{n-1}{2} \right) \cdot n,$$

which is clearly divisible by n . So

$$\frac{n(n-1)}{2} \equiv 0 \pmod{n}.$$

- If n is **even**:

Write $n = 2m$. Then

$$\frac{n(n-1)}{2} = \frac{2m(2m-1)}{2} = m(2m-1).$$

For this to be divisible by $n = 2m$, we need

$$2m \mid m(2m-1) \Rightarrow 2 \mid (2m-1),$$

which is impossible because $2m - 1$ is odd.

So:

- If n is **even**, the condition fails: **no permutation can exist**.

- If n is **odd**, the condition holds: solutions **may exist** (and we still need to **construct** one).

That's why the editorial says: "For even n — this is impossible and the answer is -1 . For odd n , one option is $p = [n, n - 1, \dots, 1]$."

So we've shown:

Necessary condition: n must be odd.

5. Constructing a solution for odd n : reversed permutation

Now we must show a **specific permutation** that satisfies the property when n is odd.

In 0-based form, consider:

$$p_i = n - 1 - i, \quad i = 0, \dots, n - 1.$$

In 1-based form this is simply:

$$[n, n - 1, \dots, 2, 1].$$

We need to show: for each $k = 0, \dots, n - 1$, in the cyclic shift by k , there is **exactly one** fixed point.

Fixed-point condition for this permutation

Again, after shift by k , element at index i goes to position $(i + k) \bmod n$. Fixed point condition:

$$(i + k) \equiv p_i \equiv n - 1 - i \pmod{n}.$$

Rearrange:

$$i + k \equiv n - 1 - i \pmod{n} \quad \Rightarrow \quad 2i \equiv n - 1 - k \pmod{n}.$$

So for each shift k , a fixed point corresponds exactly to a solution i to:

$$2i \equiv n - 1 - k \pmod{n}. \quad (\backslash *)$$

Why is there exactly one solution for each k ?

Because:

- n is **odd** $\Rightarrow \gcd(2, n) = 1$.
- Therefore, multiplication by 2 is **invertible modulo n** .

- That means the congruence

$$2i \equiv A \pmod{n}$$

always has a **unique** solution $i \pmod{n}$ for any given A .

Apply this with $A = n - 1 - k$, we see:

- For each k , there is **exactly one** i such that $(\backslash*)$ holds.
- That exact i is the (only) fixed point in the shift by k .

Hence:

For odd n , the reversed permutation $p_i = n - 1 - i$ has **exactly one fixed point in every cyclic shift**.

So it satisfies the required condition.

6. Summary of the full logic

1. Rewrite everything 0-based.
2. For fixed point after shift k :
 $(i + k) \equiv p_i \pmod{n} \Rightarrow k \equiv p_i - i \pmod{n}$.
3. "Exactly one fixed point per shift" \Rightarrow map $i \mapsto (p_i - i) \pmod{n}$ is a **bijection** onto $\{0, \dots, n - 1\}$.
4. Summing both sides:

$$\sum_{k=0}^{n-1} k = \sum_{i=0}^{n-1} (p_i - i) \pmod{n}$$

gives

$$\frac{n(n-1)}{2} \equiv 0 \pmod{n}.$$

5. This congruence holds **iff** n is **odd**. So:
 - Even n : impossible \Rightarrow answer -1 .
 - Odd n : condition OK; we still need a constructive example.
6. For odd n , choose the reversed permutation $p_i = n - 1 - i$.
7. Fixed-point condition becomes $2i \equiv n - 1 - k \pmod{n}$.
8. Since $\gcd(2, n) = 1$, this has exactly one solution i for each k .
9. Thus exactly one fixed point in each shift, done.

So the problem "**transforms to**" constructing a permutation p such that $\{(p_i - i) \pmod{n}\}$ is a complete set of residues modulo n .

For the cyclic group \mathbb{Z}_n , such a permutation exists **iff** n is odd, and the reverse permutation is one explicit solution.

7. Code (Python then C++)

We just implement exactly that:

- If n is even \rightarrow print `-1`.
- If n is odd \rightarrow print the sequence $n, n - 1, \dots, 1$.

Python

python

 Copy code

```
import sys

def solve():
    it = iter(sys.stdin.read().strip().split())
    t = int(next(it))
    out_lines = []
    for _ in range(t):
        n = int(next(it))
        if n % 2 == 0:
            out_lines.append("-1")
        else:
            # reversed permutation: n, n-1, ..., 1
            perm = " ".join(str(x) for x in range(n, 0, -1))
            out_lines.append(perm)
    sys.stdout.write("\n".join(out_lines))

if __name__ == "__main__":
    solve()
```

C++

cpp

 Copy code

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
```



```
int t;
if (!(cin >> t)) return 0;
while (t--) {
    int n;
    cin >> n;
    if (n % 2 == 0) {
        cout << -1 << '\n';
    } else {
        // reversed permutation: n, n-1, ..., 1
        for (int i = n; i >= 1; --i) {
            cout << i;
            if (i > 1) cout << ' ';
        }
        cout << '\n';
    }
}
return 0;
}
```

If you want, next we can:

- Take a small odd n (like 5 or 7),
- Write out all cyclic shifts of the reversed permutation,
- And explicitly mark the unique fixed point in each shift so you can “see” it.