Absolute permutation editorial

problem: https://www.hackerrank.com/challenges/absolute-permutation/problem

problem statement

Given N and K, a permutation of the first N natural integer is called an absolute permutation if $|a_i - i| = k$ for every $i \in [1, N]$

We must print the lexicographically smallest absolute permutation for given N and K, or -1 if no such permutation exists.

Bad solution

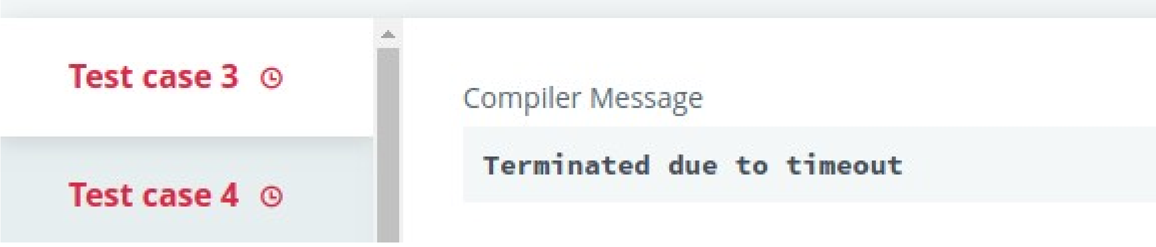
The brute force solution is to generate all possible permutation and for each permutation check if it’s a absolute permutation or not.

The time complexity for this solution is $O(N \times N!)$

C++11 code:

```
/*
  Timeout solution by brute force
  Time complexity: O(t * n * n!)
*/
#include <bits/stdc++.h>

using namespace std;

bool check(int *arr, int n, int k) {
  int i = 0;
  while (i < n && abs(arr[i] - (i+1)) == k)
    i++;

  return i >= n;
}

// Time complexity of the function: O(n * n!)
int* absolute_permutation(int *arr, int n, int k) {
  int* res = NULL;
  do {
    if (check(arr, n, k)) {
      res = arr;
      break;
    }
  }while(next_permutation(arr, arr+n));
  return res;
}
```

```
int main(){
  int t;
  cin >> t;
  cin.ignore(numeric_limits<streamsize>::max(), '\n');nk = q 2k


  for (int t_itr = 0; t_itr < t; t_itr++) {
    int n, k;
    cin >> n >> k;

    int *arr = new int[n];
    for (int i = 0 ; i < n ; ++i)
      arr[i] = i + 1;

    int* result = absolute_permutation(arr, n, k);

    if (result == NULL)
      cout << "-1";
    else {
      for (int i = 0; i < n; i++) {
        cout << result[i] <<' ';
      }
    }

    delete [] arr;

    cout << "\n";
  }

  return 0;
}
```



Compiler Message

```
Terminated due to timeout
```

Optimized solution

The question is: can we generate directly the absolute permutation, when is possible?

Firstable, we gonna check the solvability of the problem.

The solvability of the problem

Greetz to: mancha, int-e, Gilly, salt_ from ##maths on IRC freenode.

Can we generate the lexicographically smallest absolute permutation for given N and K ?

The formula of an absolute permutation:

$$\sum_{i=1}^{n} |a_i - i| - nk = 0, with\, a_i \in [1, n]$$

- if k = 0

$$\sum_{i=1}^{n} |a_i - i| = 0, <=> a_i = i\, , for\, every\, i \in [1, n]$$

- if k ≠ 0

we know that:

$$a_i - i = k\, , if\, a_i > i$$
$$a_i - i = -k\, , if\, a_i < i$$

That implies: $\sum_{i=1}^{n} (a_i - i) = 0, with\, a_i \in [1, n]$

So, we must have the same number of k 's and $-k$'s.
That leads, that n must be even.

Can we get a relation between n and k to know the solvability of the problem ?

n is even <=> $n = 2q$ <=> $nk = q2k$ <=> $n = \frac{q}{k} \times 2k$ <=> $n = m \times 2k$ <=> $2k | n$

To get an absolute permutation for given n and k :

- n **must be even**
- $2k | n$

A solution

So, we have: for every $i \in [1,n]$, $|a_i - i| = k$ <=> $a_i = i + k$ and $a_{i+k} = i$

we must be careful to not write a_i twice, so, we have to check if a_i is empty or not.

Algorithm

```
for i ∈ [1,n]
    if i is not visited
        mark i as visited
        mark i + k as visited
        $a_i = i + k$
        $a_{i+k} = i$
```

The complexity of this part of code is $O(N)$

C++ code:

```
/*
  Optimized solution
  Time complexity: O(t * n)
*/

#include <bits/stdc++.h>

using namespace std;

// Time complexity for the function: O(n)
int* absolute_permutation(int n, int k) {
  bool *visited = new bool[n];
  fill_n(visited, n, false);

  // Check solvability
  bool check = (k != 0) ? (n % (2*k) == 0) : true;

  if (!check) return NULL;
  int* res = new int[n];
  for (int i = 1 ; i <= n  ; ++i){
    if (!visited[i-1]) {
      visited[i-1] = true;
      visited[i+k-1] = true;
      res[i-1] = i + k;
      res[i+k-1] = i;
    }
  }
  return res;
}

int main(){
  int t;
  cin >> t;
  cin.ignore(numeric_limits<streamsize>::max(), '\n');

  for (int t_itr = 0; t_itr < t; t_itr++) {
    int n, k;
    cin >> n >> k;

    /*int *arr = new int[n];
    for (int i = 0 ; i < n ; ++i)
      arr[i] = i + 1;*/

    int* result = absolute_permutation(n, k);

    if (result == NULL)
      cout << "-1";
    else {
      for (int i = 0; i < n; i++) {
        cout << result[i] <<' ';
      }
    }
    cout << "\n";
  }

  return 0;
}
```