Insertion Sort

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

It is known that an array A of length N can be sorted using **insertions**:

- we consider the elements from the 1st to the N-th;
- for each element, we start swapping it with the previous one while the previous one is strictly greater than the current one.

Thus, after processing the i-th element, the prefix $A[1..i]$ becomes sorted.

Here is the pseudocode of such sorting:

```
for (i = 2; i <= n; i = i + 1) {
  j = i
  while (j > 1) and (a[j] < a[j - 1]) {
    swap(a[j], a[j - 1])
    j = j - 1
  }
}
```

Your task is, given N and K, to construct such an array A **of length** N (see details below) that when sorted by insertion sort, exactly K element swaps (calls to *swap*) will be performed.

Input

The only line contains two integers N and K separated by a space ($1 \le N \le 2 \cdot 10^5$; $0 \le K \le \frac{N \cdot (N-1)}{2}$) — the length of the required array and the required number of element swaps.

Output

In the only line, output N integers $A_1, A_2, \ldots, A_N$ separated by spaces such that:

- all $A_1, A_2, \ldots, A_N$ are **distinct** integers;
- $1 \le A_i \le N$ for all $i \in [1 \ldots N]$;
- when insertion sort described above is applied to array A, exactly K element swaps (calls to the function *swap*) will be performed.

If there are multiple possible answers, output any of them.

Examples

standard input	standard output
4 5	4 2 3 1
7 11	3 7 1 6 4 2 5

Note

Example 1.

Let us consider each iteration:

- $i = 2$:
  - $j = 2$: $a[2] < a[1]$ — we call $swap(a[2], a[1])$;
  - $j = 1$: we finish the process;
  - the array is $[2, 4, 3, 1]$.
- $i = 3$:
  - $j = 3$: $a[3] < a[2]$ — $swap(a[3], a[2])$;
  - $j = 2$: $a[2] \geq a[1]$ — we stop the process;
  - the array is $[2, 3, 4, 1]$.
- $i = 4$:
  - $j = 4$: $a[4] < a[3]$ — $swap(a[4], a[3])$;
  - $j = 3$: $a[3] < a[2]$ — $swap(a[3], a[2])$;
  - $j = 2$: $a[2] < a[1]$ — $swap(a[2], a[1])$;
  - the array is $[1, 2, 3, 4]$.

In total, $1 + 1 + 3 = 5$ calls to the function *swap* were made.

Example 2.

Let us consider each iteration:

- $i = 2$: 0 swaps were performed, the array is $[3, 7, 1, 6, 4, 2, 5]$.
- $i = 3$: 2 swaps were performed, the array is $[1, 3, 7, 6, 4, 2, 5]$.
- $i = 4$: 1 swap was performed, the array is $[1, 3, 6, 7, 4, 2, 5]$.
- $i = 5$: 2 swaps were performed, the array is $[1, 3, 4, 6, 7, 2, 5]$.
- $i = 6$: 4 swaps were performed, the array is $[1, 2, 3, 4, 6, 7, 5]$.
- $i = 7$: 2 swaps were performed, the array is $[1, 2, 3, 4, 5, 6, 7]$.

In total, $0 + 2 + 1 + 2 + 4 + 2 = 11$ calls to the function *swap* were made.