题目描述

已知连续正整数数列{K}=K1,K2,K3…Ki的各个数相加之和为S，i=N (0<S<100000, 0<N<100000), 求此数列K。

输入描述

输入包含两个参数，1）连续正整数数列和S，2）数列里数的个数N。

输出描述

如果有解输出数列K，如果无解输出-1。

用例

输入	525 6
输出	85 86 87 88 89 90
说明	无

输入	3 5
输出	-1
说明	无

题目解析

本题解题思路：

由于要求连续正整数数列的和，因此，S / N 的结果必然是数列的中间值，比如 525 / 6 = 87.5，由于6是偶数个，因此87.5其实就是 87 和 88 的中间值。

再比如 9 / 3 = 3，而9是 2 3 4 数列的和，而3是奇数个，因此3就是2 3 4 数列的中间值。

得到中间值后，我们就可以根据连续正整数数列半径求出连续数列两个边界值，如果左边界<=0，那么就直接返回-1，否则求得边界就是符合要求的。

最后，返回两个边界之内的值即可。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const [sum, n] = line.split(" ").map(Number);
  console.log(getResult(sum, n));
});

function getResult(sum, n) {
  let left, right;

  if (n % 2 == 0) {
    const halfLen = n / 2 - 1;
    left = Math.floor(sum / n) - halfLen;
    right = Math.ceil(sum / n) + halfLen;
  } else {
    const mid = sum / n;
    const halfLen = Math.floor(n / 2);
    left = mid - halfLen;
    right = mid + halfLen;
  }

  if (left <= 0) return -1;

  const arr = [];
  for (let i = left; i <= right; i++) arr.push(i);

  return arr.join(" ");
}
```

Java算法源码

```java
import java.util.Scanner;
import java.util.StringJoiner;

public class Main {
  // 输入获取
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int sum = sc.nextInt();
    int n = sc.nextInt();

    System.out.println(getResult(sum, n));
  }

  // 算法入口
  public static String getResult(int sum, int n) {
    int left, right;

    if (n % 2 == 0) {
      int halfLen = n / 2 - 1;
      left = sum / n - halfLen;
      right = sum / n + 1 + halfLen;
    } else {
      int mid = sum / n;
      int halfLen = n / 2;
      left = mid - halfLen;
      right = mid + halfLen;
    }

    if (left < 0) return "-1";

    StringJoiner sj = new StringJoiner(" ");
    for (int i = left; i <= right; i++) sj.add(i + "");
    return sj.toString();
  }
}
```

Python算法源码

```python
# 输入获取
sumV, n = map(int, input().split())


# 算法入口
def getResult():
    left, right = -1, -1

    if n % 2 == 0:
        halfLen = n // 2 - 1
        left = sumV // n - halfLen
        right = sumV // n + 1 + halfLen
    else:
        mid = sumV // n
        halfLen = n // 2
        left = mid - halfLen
        right = mid + halfLen

    if left <= 0:
        return -1

    return " ".join([str(i) for i in range(left, right + 1)])


# 算法调用
print(getResult())
```