1652. Defuse the Bomb

You have a bomb to defuse, and your time is running out! Your informer will provide you with a **circular** array `code` of length of `n` and a key `k`.

To decrypt the code, you must replace every number. All the numbers are replaced **simultaneously**.

- If `k > 0`, replace the `i`th number with the sum of the **next** `k` numbers.
- If `k < 0`, replace the `i`th number with the sum of the **previous** `k` numbers.
- If `k == 0`, replace the `i`th number with `0`.

As `code` is circular, the next element of `code[n-1]` is `code[0]`, and the previous element of `code[0]` is `code[n-1]`.

Given the **circular** array `code` and an integer key `k`, return *the decrypted code to defuse the bomb*!

Example 1:

```
Input: code = [5,7,1,4], k = 3
Output: [12,10,16,13]
Explanation: Each number is replaced by the sum of the next 3 numbers. The decrypted code is [7+1+4, 1+4+5, 4+5+7, 5+7+1]. Notice that the numbers wrap around.
```

Example 2:

```
Input: code = [1,2,3,4], k = 0
Output: [0,0,0,0]
Explanation: When k is zero, the numbers are replaced by 0.
```

Example 3:

```
Input: code = [2,4,9,3], k = -2
Output: [12,5,6,13]
Explanation: The decrypted code is [3+9, 2+3, 4+2, 9+4]. Notice that the numbers wrap around again. If k is negative, the sum is of the previous numbers.
```

Constraints:

- `n == code.length`
- `1 <= n <= 100`
- `1 <= code[i] <= 100`
- `-(n - 1) <= k <= n - 1`

1652. Defuse the Bomb

```
/*
   Brute force
   Time complexity: O(n×|(k)|)=O(n^2), if k=n
   Space complexity: O(1)
*/
typedef std::vector<int> vi;
class Solution{
   public:
      vi decrypt(vi& code, int k){
         int n=code.size();

         vi ans(n,0);

         if(k==0) return ans;


         // k>0 => dir=1 (Go right)
         // k<0 => dir=-1 (Go left)
         int dir=(k>0)?1:-1;

         // For each index i (0<=i<=n-1)
         for(int i=0;i<n;++i){
            // compute the sum of next/previous k elements
            int s=0;
            for(int j=1;j<=std::abs(k);++j){
               // Because the array is circular
               // we need to get the correct index of the
               // element to add to the sum
               // if k>0 (dir=1), shift the index i to the right by j positions
               // if k<0 (dir=-1), shift the index i to the left by j positions
               int idx=((((dir*j)+i)%n)+n)%n;
               s+=code[idx];
            }
            ans[i]=s;
         }

         return ans;
      }
};
```

1652. Defuse the Bomb

```
/*
   Prefix sums
   Time complexity: O(2n+2n+2n+2n+n)=O(9n)=O(n)
   Space complexity: O(2n+2n)=O(4n)=O(n)
*/
typedef std::vector<int> vi;
class Solution {
   public:
      vi decrypt(vi& code, int k){
         int n=code.size();

         vi ans(n,0);

         if(k==0) return ans;

         // Maintain circular property
         int m=2*n;
         vi arr(m);
         for(int i=0;i<m;++i) arr[i]=code[i%n];

         // Preprocess prefix sums
         vi prefix_sum(m+1);
         prefix_sum[0]=0;
         for(int i=1;i<=m;++i) prefix_sum[i]=prefix_sum[i-1]+arr[i-1];

         // Get answers
         for(int i=0;i<n;++i){
            ans[i]=k>0?prefix_sum[i+k+1]-prefix_sum[i+1]:prefix_sum[n+i]-prefix_sum[n+i-abs(k)];
         }

         return ans;
      }
};
```

1652. Defuse the Bomb

```
/*
   Sliding window
   Time complexity: O(k+n)=O(2n), if k=n => O(n)
   Space complexity: O(1)
*/
typedef std::vector<int> vi;
class Solution {
   public:
      int mod(int a,int b){
         return ((a%b)+b)%b;
      }

      vi decrypt(vi& code, int k){
         int n=code.size();

         vi ans(n,0);

         if(k==0) return ans;

         // k>0 => dir=1 (Go right)
         // k<0 => dir=-1 (Go left)
         int dir=(k>0)?1:-1;

         // Create a window of size k
         // if k>0, create window from left to right
         // if k<0, create window from right to left
         int win=0;
         for(int j=1;j<=std::abs(k);++j){
            int idx=mod(dir*j,n);
            win+=code[idx];
         }

         // Window start and end indices
         int start=mod(dir,n);
         int end=mod(k,n);
         if(start>end) std::swap(start,end);
```

```
        int i=0;
        while(i<n){
            ans[i]=win;

            // Slide the window
            win-=code[start];
            win+=code[mod(end+1,n)];
            start=mod(start+1,n);
            end=mod(end+1,n);

            i++;
        }

        return ans;
    }
};
```