3208. Alternating Groups II

There is a circle of red and blue tiles. You are given an array of integers `colors` and an integer `k`. The color of tile `i` is represented by `colors[i]`:

- `colors[i] == 0` means that tile `i` is **red**.
- `colors[i] == 1` means that tile `i` is **blue**.

An **alternating** group is every `k` contiguous tiles in the circle with **alternating** colors (each tile in the group except the first and last one has a different color from its **left** and **right** tiles).

Return the number of **alternating** groups.

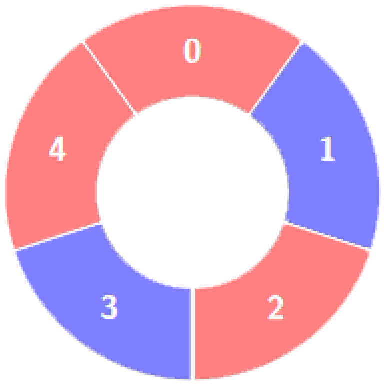
Note that since `colors` represents a **circle**, the **first** and the **last** tiles are considered to be next to each other.

Example 1:

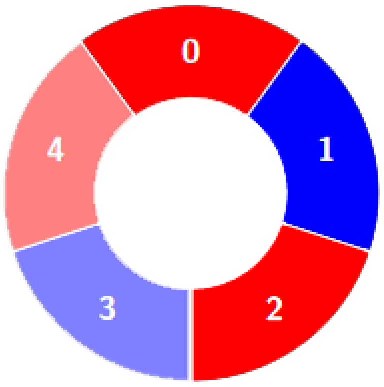
Input: colors = [0,1,0,1,0], k = 3

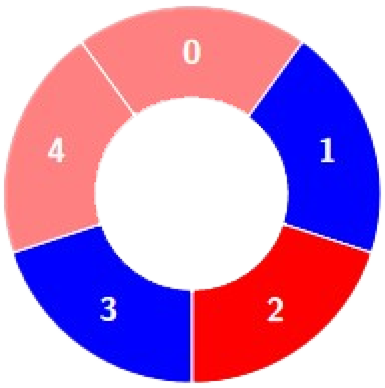
Output: 3

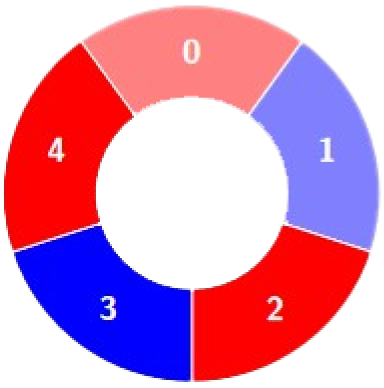
Explanation:



Alternating groups:





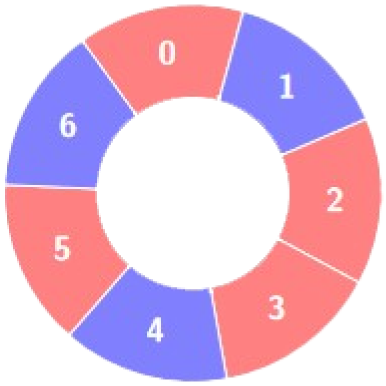


Example 2:

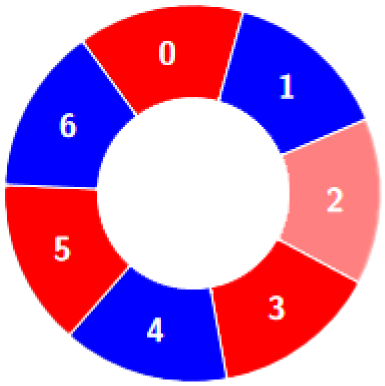
Input: colors = [0,1,0,0,1,0,1], k = 6

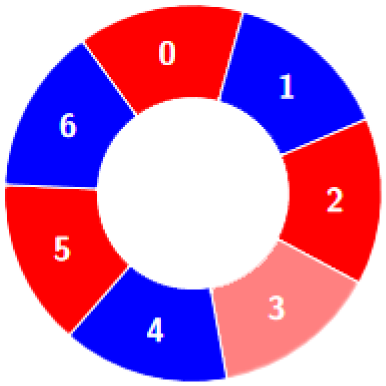
Output: 2

Explanation:



Alternating groups:



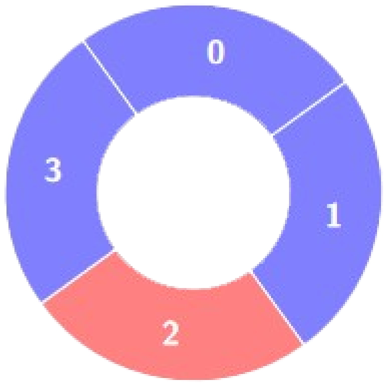


Example 3:

Input: colors = [1,1,0,1], k = 4

Output: 0

Explanation:



3208. Alternating Groups II

```
/*
   Expand array+Dynamic sliding window
   Time complexity: O(2n)
   Space complexity:O(n)
*/

class Solution {
public:
   int numberOfAlternatingGroups(std::vector<int>& colors, int k){
      // Expand the given by the elements in range[0,k-2]
      // to simulate the circularity
      for(int i=0;i<k-1;++i) colors.push_back(colors[i]);

      int n=colors.size();

      int ans=0;

      int i=0; // start of the window
      int j=0; // end of the window

      while(j<n-1){
         // If the last tile in the actual window have the same
         // color with the next tile
         if(colors[j]==colors[j+1]){
            // Start new window from the next tile
            i=j+1;
            j=i;
            continue;
         }

         // Otherwise, expand the window
         j++;

         // If the length of the window reaches k
         if(j-i+1==k){
            ans++; // Count the current window in the answer
            i++; // Shrink the window from the left
         }
      }

      return ans;
   }
};
```

3208. Alternating Groups II

```
/*
    Circular array+Dynamic sliding window
    Time complexity: O(2n)
    Space complexity:O(1)
*/

class Solution {
public:
    int numberOfAlternatingGroups(std::vector<int>& colors, int k){
        int n=colors.size();

        int ans=0;

        int i = 0; // start of the window
        int j = 0; // end of the window

        // Expand the size to simulate circular array
        while(j<n+k-2){
            if(colors[j%n]==colors[(j+1)%n]){
                i=j+1;
                j=i;
                continue;
            }

            j++;

            if(j-i+1==k){
                ans++;
                i++;
            }
        }

        return ans;
    }
};
```