小团需要购买 m 样装饰物。商店出售 n 种装饰物，按照从小到大的顺序从左到右摆了一排。对于每一个装饰物，小团都给予了一个美丽值。
小团希望购买的装饰物有着相似的大小，所以他要求购买的装饰物在商店中摆放的位置是连续的一段。小团还认为，一个装饰物的美丽值不能低于 k，否则会不好看。
现在，请你计算小团有多少种不同的购买方案。
小团需要购买 m 样装饰物。商店出售 n 种装饰物，按照从小到大的顺序从左到右摆了一排。对于每一个装饰物，小团都给予了一个美丽值。
小团希望购买的装饰物有着相似的大小，所以他要求购买的装饰物在商店中摆放的位置是连续的一段。小团还认为，一个装饰物的美丽值不能低于 k，否则会不好看。
现在，请你计算小团有多少种不同的购买方案。

```
public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String[] params = br.readLine().trim().split(" ");
        int n = Integer.parseInt(params[0]);
        int m = Integer.parseInt(params[1]);
        int k = Integer.parseInt(params[2]);
        params = br.readLine().trim().split(" ");
        int[] pretty = new int[n];
        for(int i = 0; i < n; i++)
            pretty[i] = Integer.parseInt(params[i]);
        int count = 0;
        // 构建一个小根堆作为窗口
        PriorityQueue<Integer> window = new PriorityQueue<>();
        // 滑窗遍历
        for(int left = 0; left <= n - m; left++){
            if(left == 0){
                // 构建初始窗口
                for(int i = left; i <= left + m - 1; i++)
                    window.offer(pretty[i]);
            }else{
                window.remove(pretty[left - 1]);
                window.offer(pretty[left + m - 1]);
            }
            // 这一段连续区间满足要求
            if(window.peek() >= k) count++;
        }
        System.out.println(count);
    }
}
```

小团和小美正在玩一个填数游戏，这个游戏是给一个等式，其中有一些数被挖掉了，你需要向其中填数字，使得等式成立。

比如 ___+12=34，那么横线填的一定是 22
现在，这个游戏到了最后一关，这一关的等式很奇特：_+_+_+...+_=n
这里可以填任意多个正整数（甚至可能是 1 个），只要这些数的和等于 n 即可。
但是，有一个额外的限制，填入的所有数必须小于等于 k，大于等于 1，填入的数的最大值必须大于等于 d。

请你计算，有多少个不同的等式满足这些限制。由于答案可能很大，请将答案 mod(998244353)后输出。

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String[] params = br.readLine().trim().split(" ");
        int n = Integer.parseInt(params[0]);
        int k = Integer.parseInt(params[1]);
        int d = Integer.parseInt(params[2]);
        // dp[i][0]表示凑成 i 时没有用大于等于 d 的方案数，dp[i][1]表示凑成 i 时用了大于等于 d 的方案数
        long[][] dp = new long[n + 1][2];
        dp[0][0] = 1;
        for(int i = 1; i <= n; i++){
            // 题目要求填的正整数范围是从 1~k，但是也不能超过目标值 i
            for(int j = 1; j <= Math.min(k, i); j++){
                if(j < d){
                    dp[i][0] = (dp[i][0] + dp[i - j][0]) % 998244353;
                    dp[i][1] = (dp[i][1] + dp[i - j][1]) % 998244353;
                }else
                    dp[i][1] = (dp[i][1] + dp[i - j][0] + dp[i - j][1]) % 998244353;
            }
        }
        // 输出凑出了 n 并且使用了大于等于 d 的数的方案数
        System.out.println(dp[n][1]);
    }
}
```