题目描述：

在做物理实验时，为了计算物体移动的速率，通过相机等工具周期性的采样物体移动距离。由于工具故障，采样数据存在误差甚至错误的情况。需要通过一个算法过滤掉不正确的采样值。不同工具的故障模式存在差异，算法的各类门限会根据工具类型做相应的调整。请实现一个算法，计算出给定一组采样值中正常值的最长连续周期。

判断第 *i* 个周期的采样数据 *S[i]*是否正确的规则如下（假定物体移动速率不超过 *10* 个单元，

前一个采样周期 *S[i-1]*）：

- *S[i]* <= *0*，即为错误值
- *S[i]* < *S[i-1]*，即为错误值
- *S[i]* - *S[i-1]* >= *10*，即为错误值
- 其它情况为正常值

判断工具是否故障的规则如下：

- 在 *M* 个周期内，采样数据为错误值的次数为 *T*（次数可以不连续），则工具故障。

判断故障恢复的条件如下：

- 产生故障后的 *P* 个周期内，采样数据一直为正常值，则故障恢复。

错误采样数据的处理方式：

- 检测到故障后，丢弃从故障开始到故障恢复的采样数据。
- 在检测到工具故障之前，错误的采样数据，则由最近一个正常值代替；如果前面没有正常的采样值，则丢弃此采样数据。

给定一段周期的采样数据列表 *S*，计算正常值的最长连续周期。

输入描述：

故障确认周期数和故障次数门限分别为 *M* 和 *T*，故障恢复周期数为 *P*。

第 *i* 个周期，检测点的状态为 *Si*

输入为两行，格式如下：

M T P

S1 S2 S3 ...

M、T 和 P 的取值范围为[1, 100000]

Si 取值范围为[0, 100000]，i 从 0 开始编号

输出描述：

一行输出正常值的最长连续周期

补充说明：

示例 1

输入：

```
10 6 3

-1 1 2 3 100 10 13 9 10
```

输出：

```
8
```

说明：

S[0]，S[4]，S[7]，S[8]为错误值。S[0]之前没有正常的采样数据，丢弃 S[0]。S[4]和 S[7]不满足故障条件，此值分别由 S[3]和 S[6]代替，即 S[4]为 3，S[7]为 13。替换后，S[8]小于 S[7]，也是错误值。

示例 2

输入：

```
5 3 3

0 1 2 -1 4 3 6 7 6 6 10 11 12
```

输出：

```
9
```

说明：

S[3]，S[5]，S[8]，S[9]为错误值。从 S[3]到 S[7]的 5 个周期内只有两个错误值 S[3]和 S[5]。从 S[5]到 S[9]的 5 个周期内有三个错误值 S[5]、S[8]和 S[9]，工具故障。丢弃 S[9]到 S[12]的值。

示例 3

输入：

```
5 3 3
1 2 -1 -2 -3 6 7 8 9 10 11 12
```

输出：

```
5
```

说明：

S[2]，S[3]，S[4]为错误值。从 S[2]到 S[6]的 5 个周期内有三个错误值，工具故障。丢弃 S[4]到 S[6]的值。有两段正常连续周期，S[0]到 S[3]（周期数为 4）和 S[7]到 S[11]（周期数为 5）。

```
#include <bits/stdc++.h>
#include <sstream>
#include <string>
#include <vector>
using namespace std;

int M, T, P;
string str;
bool ok(int pre, int cur) {
    if(pre == -1) {
        if(cur <= 0 ) return 0;
        return 1;
    }
    if(cur <= 0 || pre > cur || cur - pre > 10) return 0;
    return 1;
}
```

```
int main() {
	cin >> M;
	cin >> T;
	cin >> P;
	getline(cin, str);
	getline(cin, str);
	stringstream ss(str);
	string item;
	vector<int> a, b;
	while (getline(ss, item, ' ')) {
		if(!item.empty()) {
			a.push_back(stoi(item));
			b.push_back(0);
		}
	}
	int n = a.size();
	for (int i = 0; i < n; i++) {
		int pre = -1;
		if (i > 0 && a[i-1] > 0) pre = a[i-1];
		if(!ok(pre, a[i])) {
			b[i] = 1;
			if(i > 0) a[i] = a[-1];
		}
	}

	queue<int> que;
	int st = 0, qs = 0, error = 0, ans = 0, f = 1, cur = 0, cs = 0;
	while (st < n && b[st]) {
		que.push(st++);
		qs++;
		error++;
	}
	while (qs > M) {
		que.pop();
		qs--;
	}
	if(error >= T) f = 0;
	if(qs == M ) que.pop(), qs--,error--;
	for (int i = st; i < n; i++) {
		if (f) {
			que.push(i);
			qs++;
			if(b[i]) error++;
			if(qs == M && error >= T){
```

```
                f = 0,cur = 0;
            }
            else {
                cur++;
                ans = max(ans, cur);
            }
            if(qs >= M){
                if(b[que.front()]) error--;
                que.pop();
            }
        }else {
            if(!b[i]) cs++;
            else cs = 0;
            if(cs == P){
                f = 1;
                cur = 1;
                ans = max(ans, cur);
                que.push(i);
                qs++;
            }
        }

    }
    cout << ans << endl;
    return 0;
}
```