五子棋迷题目描述：

张兵和王武是五子棋迷，工作之余经常切磋棋艺。这不，这会儿又下起来了。走了一会儿，轮张兵了，对着一条线思考起来了，这条线上的棋子分布如下:

用数组表示: -1 0 1 1 1 0 1 0 1 -1

棋子分布说明:

1. -1 代表白子，0 代表空位，1 代表黑子

2. 数组长度 L，满足 1 < L < 40，且 L 为奇数

你得帮他写一个程序，算出最有利的出子位置。 最有利定义:

1. 找到一个空位(0)，用棋子(1/-1)填充该位置，可以使得当前子的最大连续长度变大;

2. 如果存在多个位置，返回最靠近中间的较小的那个坐标;

3. 如果不存在可行位置，直接返回-1;

4. 连续长度不能超过 5 个(五字棋约束);

输入描述：

第一行: 当前出子颜色

第二行: 当前的棋局状态

输出描述：

1 个整数，表示出子位置的数组下标

补充说明：

示例 1

输入：

```
1

-1 0 1 1 1 0 1 0 1 -1 1
```

输出：

```
5
```

说明：

当前为黑子（1），放置在下标为 5 的位置，黑子的最大连续长度，可以由 3 到 5

示例 2

输入：

```
-1

-1 0 1 1 1 0 1 0 1 -1 1
```

输出：

```
1
```

说明：

当前为白子，唯一可以放置的位置下标为 1， 白子的最大长度，由 1 变为 2

示例 3

输入：

```
1
0 0 0 0 1 0 0 0 0 1 0
```

输出：

```
5
```

说明：

可行的位置很多，5 最接近中间的位置坐标

```
#include <iostream>
```

```
#include <string>

#include <map>

#include <vector>

#include <queue>

#include <cmath>

#include <algorithm>

#include <utility>

#include <random>


#define DEBUG


int main() {

    int color;

    std::vector<int> list;


    std::cin >> color;

    std::cin.ignore();


    std::string str;

    std::getline(std::cin, str);

//    std::cout << str << std::endl;

    str.push_back(' ');
```

```
    int head = 0;
    for (int i = 0; i < str.size(); i++) {
        if (str[i] == ' ') {
            std::string s(str, head, i - head);
            if (!s.empty()) {
                list.push_back(std::stoi(s));
            }
            head = i + 1;
        }
    }

    for (int i = 0; i < list.size(); i++) {

    }

//    for (auto n : list) {
//        std::cout << n << " ";
//    }
//    std::cout << std::endl;

    int max_count = 0;
    int max_count_1 = 0;
```

```
int cur_count = 0;
int cur_count_1 = 0;
int nega_count = 0;
int pos = -1;
for (int i = 0; i < list.size(); i++) {
    if (list[i] == color) {
        cur_count_1++;
        if (cur_count_1 > max_count_1) {
            max_count_1 = cur_count_1;
        }
        cur_count++;
    } else if (list[i] == -color) {
        cur_count = 0;
        cur_count_1 = 0;
    } else {
        cur_count++;
        for (int j = i + 1; j < list.size(); j++) {
            if (list[j] == color) {
                cur_count++;
            } else {
                break;
            }
```

```
            }
            if (cur_count > max_count && cur_count <= 5) {
                pos = i;
                max_count = cur_count;
            } else if (cur_count == max_count) {
                if (std::abs(pos - (int)(list.size()/2)) > std::abs(i - (int)(list.size()/2))) {
                    pos = i;
                } else if (std::abs(pos - (int)(list.size()/2)) == std::abs(i - (int)(list.size()/2))) {
                    pos = pos < i ? pos : i;
                }
                max_count = cur_count;
            }
            cur_count = 0;
            cur_count_1 = 0;
        }
    }

//    std::cout << max_count_1 << ", " << max_count << std::endl;

    if (max_count_1 > max_count) {
```

```
        std::cout << -1 << std::endl;
    } else {
        std::cout << pos << std::endl;
    }


    return 0;
}
```