找车位

题目描述：

停车场有一横排车位，0 代表没有停车，1 代表有车。至少停了一辆车在车位上，也至少有一个空位没有停车。

为了防剐蹭，需为停车人找到一个车位，使得距停车人的车最近的车辆的距离是最大的，返回此时的最大距离。

输入描述：

1、一个用半角逗号分割的停车标识字符串，停车标识为 0 或 1，0 为空位，1 为已停车。

2、停车位最多 100 个。

输出描述：

输出一个整数记录最大距离。

示例 1

输入：

```
1,0,0,0,0,1,0,0,1,0,1
```

输出：

```
2
```

说明：

当车停在第 3 个位置上时，离其最近的的车距离为 2（1 到 3）。
当车停在第 4 个位置上时，离其最近的的车距离为 2（4 到 6）。
其他位置距离为 1。
因此最大距离为 2。

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
using namespace std;

void split(vector<char>& chVec, const string& str, char ch){
    int offset = 0;
    int idx =0;
    while( (idx=str.find(ch, offset)) != string::npos ){
        chVec.push_back(str.substr(offset, idx)[0]);
        offset = idx+1;
    }
    chVec.push_back(str.substr(offset)[0]);
}

int maxDistance(const string& str){
    vector<char> stops;
```

```
    split(stops, str, ',');

    vector<int> vec;
    for (int i =0; i < stops.size(); ++i){
        if (stops[i]=='1'){
            vec.push_back(i);
        }
    }

    int res = 0;
    if (vec[0] != 0){
        res = vec[0];
    }

    for (int i = 0; i < vec.size()-1; ++i){
        int d = vec[i+1] - vec[i]-1;
        res = max(res, d / 2 + d%2);
    }

    if (vec[vec.size()-1] != stops.size()-1){//若末尾不是'1'
        int d = stops.size()-1 - vec[vec.size()-1];
        res = max(res, d);
    }
    return res;
}

int main()
{
    string str;
    cin >> str;
    cout << maxDistance(str) << endl;
    return 0;
}
```