

---

## LeetCode Notebook

Ruohong Jiao  
*NUAA*

2021 年 1 月 7 日

# 目录

<b>附录 A Problem List</b>	<b>2</b>
A.1 Leetcode 1 . . . . .	2
A.2 Leetcode 2 . . . . .	2
A.3 Leetcode 3 . . . . .	2
A.4 Leetcode 6 . . . . .	3
A.5 Leetcode 11 . . . . .	3
A.6 Leetcode 15 . . . . .	4
A.7 Leetcode 16 . . . . .	4
A.8 Leetcode 19 . . . . .	5
A.9 Leetcode 25 . . . . .	5
A.10 Leetcode 26 . . . . .	5
A.11 Leetcode 42 . . . . .	6
A.12 Leetcode 56 . . . . .	7
A.13 Leetcode 61 . . . . .	7
A.14 Leetcode 121 . . . . .	7
A.15 Leetcode 138 . . . . .	8
A.16 Leetcode 141 . . . . .	9
A.17 Leetcode 202 . . . . .	9
A.18 Leetcode 206 . . . . .	10
A.19 Leetcode 209 . . . . .	10
A.20 Leetcode 387 . . . . .	10
A.21 Leetcode 876 . . . . .	11
<b>附录 B Code List</b>	<b>11</b>
B.1 Leetcode 1 . . . . .	11
B.2 Leetcode 2 . . . . .	13
B.3 Leetcode 3 . . . . .	16
B.4 Leetcode 6 . . . . .	18
B.5 Leetcode 11 . . . . .	21
B.6 Leetcode 15 . . . . .	23
B.7 Leetcode 16 . . . . .	26
B.8 Leetcode 19 . . . . .	28
B.9 Leetcode 25 . . . . .	31
B.10 Leetcode 26 . . . . .	34
B.11 Leetcode 42 . . . . .	36
B.12 Leetcode 56 . . . . .	38
B.13 Leetcode 61 . . . . .	40
B.14 Leetcode 121 . . . . .	43
B.15 Leetcode 138 . . . . .	45
B.16 Leetcode 141 . . . . .	49
B.17 Leetcode 202 . . . . .	51
B.18 Leetcode 206 . . . . .	53
B.19 Leetcode 209 . . . . .	56
B.20 Leetcode 387 . . . . .	58
B.21 Leetcode 876 . . . . .	60

## 附录 A Problem List

### A.1 Leetcode 1

#### Problem Description:

##### 两数之和

给定一个整数数组 *nums* 和一个目标值 *target*，请你在该数组中找出和为目标值的那两个整数，并返回他们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素不能使用两遍。

#### Sample:

input:

```
1
2 给定 nums = [2, 7, 11, 15], target = 9
```

output:

```
1
2 因为 nums[0] + nums[1] = 2 + 7 = 9
3 所以返回 [0, 1]
```

#### Solution (Codes at B.1):

没有说明输入数字一定是正整数，不能先排序后提取小于 *target* 的数进行求解。

懒一点， $n^2$  循环。勤快一点用红黑树、堆进行存储然后查询  $n\log n$ 。

### A.2 Leetcode 2

#### Problem Description:

##### 两数相加

给出两个非空的链表用来表示两个非负的整数。其中，它们各自的位数是按照逆序的方式存储的，并且它们的每个节点只能存储一位数字。

如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。

您可以假设除了数字 0 之外，这两个数都不会以 0 开头。

#### Sample:

input:

```
1
2 输入: (2 -> 4 -> 3) + (5 -> 6 -> 4)
```

output:

```
1
2 输出: 7 -> 0 -> 8
3 原因: 342 + 465 = 807
```

#### Solution (Codes at B.2):

大数加法，注意指针/引用转换。

### A.3 Leetcode 3

#### Problem Description:

##### 无重复字符的最长子串

给定一个字符串，请你找出其中不含有重复字符的最长子串的长度。

#### Sample:

input:

```
1
2 输入: "abcabcbb"
3
```

```
4 输入: "bbbb"
5
6 输入: "pwwkew"
```

output:

```
1
2 输出: 3
3 解释: 因为无重复字符的最长子串是 "abc", 所以其长度为 3。
4
5 输出: 1
6 解释: 因为无重复字符的最长子串是 "b", 所以其长度为 1。
7
8 输出: 3
9 解释: 因为无重复字符的最长子串是 "wke", 所以其长度为 3。
10  请注意, 你的答案必须是 子串 的长度, "pwke" 是一个子序列, 不是子串。
```

**Solution** (Codes at B.3):

标记 l 和 r, 遍历一遍, 更新答案。时间方面, 能用数组不用 STL。

## A.4 Leetcode 6

**Problem Description:**

**Z 字形变换**

将一个给定字符串根据给定的行数, 以从上往下、从左到右进行 Z 字形排列。

比如输入字符串为"LEETCODEISHIRING" 行数为 3 时, 排列如下:

```
1 L   C   I   R
2 E T O E S I I G
3 E   D   H   N
```

之后, 你的输出需要从左往右逐行读取, 产生出一个新的字符串, 比如: "LCIRETOESIIGEDHN"。

**Sample:**

input:

```
1
2 输入: s = "LEETCODEISHIRING", numRows = 3
3
4 输入: s = "LEETCODEISHIRING", numRows = 4
```

output:

```
1
2 输出: "LCIRETOESIIGEDHN"
3
4 输出: "LDREOEIIECIHNTSG"
5 解释:
6
7 L       D       R
8 E   O E   I I
9 E C   I H   N
10 T       S       G
```

**Solution** (Codes at B.4):

计算第一行字符 id, 前后搜索。

## A.5 Leetcode 11

**Problem Description:**

**盛最多水的容器**

给你  $n$  个非负整数  $a_1, a_2, \dots, a_n$ , 每个数代表坐标中的一个点  $(i, a_i)$ 。在坐标内画  $n$  条垂直线, 垂直线  $i$  的两个端点分别为  $(i, a_i)$  和  $(i, 0)$ 。找出其中的两条线, 使得它们与  $x$  轴共同构成的容器可以容纳最多的水。

说明: 你不能倾斜容器, 且  $n$  的值至少为 2。

**Sample:**

input:

```
1
2 输入: [1,8,6,2,5,4,8,3,7]
```

otput:

```
1
2 输出: 49
```

**Solution** (Codes at B.5):

两边到中间遍历一遍，贪心选择更高的边维持原状。

**A.6 Leetcode 15****Problem Description:****三数之和**

给你一个包含  $n$  个整数的数组 `nums`，判断 `nums` 中是否存在三个元素  $a, b, c$ ，使得  $a + b + c = 0$ ？请你找出所有满足条件且不重复的三元组。

注意：答案中不可以包含重复的三元组。

**Sample:**

input:

```
1
2 给定数组 nums = [-1, 0, 1, 2, -1, -4],
```

otput:

```
1
2 满足要求的三元组集合为:
3 [
4     [-1, 0, 1],
5     [-1, -1, 2]
6 ]
```

**Solution** (Codes at B.6):

不要依赖 STL，不是所有题目都需要离散化处理。对于一般的链表问题，去重遍历就可以达到离散化的效果。先排序，指定一个数值的基础上再挑选符合条件的另外两个数值。贪心匹配两个数值的大小。

**A.7 Leetcode 16****Problem Description:****最接近的三数之和**

给定一个包括  $n$  个整数的数组 `nums` 和一个目标值 `target`。找出 `nums` 中的三个整数，使得它们的和与 `target` 最接近。返回这三个数的和。假定每组输入只存在唯一答案。

提示：

$$3 \leq \text{nums.length} \leq 10^3$$

$$-10^3 \leq \text{nums}[i] \leq 10^3$$

$$-10^4 \leq \text{target} \leq 10^4$$

**Sample:**

input:

```
1
2 输入: nums = [-1,2,1,-4], target = 1
```

otput:

```
1
2 输出：2
3 解释：与 target 最接近的和是 2 (-1 + 2 + 1 = 2)。
```

**Solution** (Codes at B.7):

不要依赖 STL，不是所有题目都需要离散化处理。对于一般的链表问题，去重遍历就可以达到离散化的效果。先排序，指定一个数值的基础上再挑选符合条件的另外两个数值。贪心匹配两个数值的大小。

## A.8 Leetcode 19

**Problem Description:**

**Problem Description:**

**删除链表的倒数第 N 个节点**

给定一个链表，删除链表的倒数第 n 个节点，并且返回链表的头结点。

**Sample:**

input:

```
1
2 给定一个链表：1->2->3->4->5，和 n = 2。
```

output:

```
1
2 当删除了倒数第二个节点后，链表变为 1->2->3->5。
```

**Solution** (Codes at B.8):

递归记录个数。

## A.9 Leetcode 25

**Problem Description:**

**Problem Description:**

**K 个一组翻转链表**

给你一个链表，每 k 个节点一组进行翻转，请你返回翻转后的链表。

k 是一个正整数，它的值小于或等于链表的长度。

如果节点总数不是 k 的整数倍，那么请将最后剩余的节点保持原有顺序。

说明：

你的算法只能使用常数的额外空间。

你不能只是单纯的改变节点内部的值，而是需要实际进行节点交换。

**Sample:**

input:

```
1
2 给你这个链表：1->2->3->4->5
```

output:

```
1
2 当 k = 2 时，应当返回：2->1->4->3->5
3
4 当 k = 3 时，应当返回：3->2->1->4->5
```

**Solution** (Codes at B.9):

标记子链表首尾，化简为链表反转问题，递归解决。

## A.10 Leetcode 26

**Problem Description:**

### 删除排序数组中的重复项

给定一个排序数组，你需要在原地删除重复出现的元素，使得每个元素只出现一次，返回移除后数组的新长度。

不要使用额外的数组空间，你必须在原地修改输入数组并在使用  $O(1)$  额外空间的条件下完成。

说明:

为什么返回数值是整数，但输出的答案是数组呢?

请注意，输入数组是以引用方式传递的，这意味着在函数里修改输入数组对于调用者是可见的。

你可以想象内部操作如下:

```
1 输入: nums = [-1,2,1,-4], target = 1
2
3 // nums 是以“引用”方式传递的。也就是说，不对实参做任何拷贝
4 int len = removeDuplicates(nums);
5
6 // 在函数里修改输入数组对于调用者是可见的。
7 // 根据你的函数返回的长度，它会打印出数组中该长度范围内的所有元素。
8 for (int i = 0; i < len; i++) {
9     print(nums[i]);
10 }
```

**Sample:**

input:

```
1
2 给定数组 nums = [1,1,2],
3
4 给定 nums = [0,0,1,1,1,2,2,3,3,4],
```

output:

```
1
2 函数应该返回新的长度 2，并且原数组 nums 的前两个元素被修改为 1，2。
3 你不需要考虑数组中超出新长度后面的元素。
4
5 函数应该返回新的长度 5，并且原数组 nums 的前五个元素被修改为 0，1，2，3，4。
6 你不需要考虑数组中超出新长度后面的元素。
```

**Solution** (Codes at B.10):

原始数组已经排过序了。注意当数据量大的时候，判断条件越少越好。

## A.11 Leetcode 42

**Problem Description:**

**接雨水**

给定  $n$  个非负整数表示每个宽度为 1 的柱子的高度图，计算按此排列的柱子，下雨之后能接多少雨水。

上面是由数组  $[0,1,0,2,1,0,1,3,2,1,2,1]$  表示的高度图，在这种情况下，可以接 6 个单位的雨水（蓝色部分表示雨水）。

**Sample:**

input:

```
1
2 输入: [0,1,0,2,1,0,1,3,2,1,2,1]
```

output:

```
1
2 输出: 6
```

**Solution** (Codes at B.11):

左右打表记录最大值，查询之后得出结果。

老问题，注意当数据量大的时候，判断条件越少越好。以及，多用 C++11 的新初始化特性可以减少时间开销。

## A.12 Leetcode 56

### Problem Description:

#### 合并区间

给出一个区间的集合，请合并所有重叠的区间。

### Sample:

input:

```
1
2 输入: intervals = [[1,3],[2,6],[8,10],[15,18]]
3
4 输入: intervals = [[1,4],[4,5]]
```

otput:

```
1
2 输出: [[1,6],[8,10],[15,18]]
3 解释: 区间 [1,3] 和 [2,6] 重叠, 将它们合并为 [1,6].
4
5 输出: [[1,5]]
6 解释: 区间 [1,4] 和 [4,5] 可被视为重叠区间。
```

### Solution (Codes at B.12):

自定义排序之后合并。需要注意对是 C++ 快排默认比较顺序是挨个从小到大，如果可以使用原始比较函数就不要自己重新写，还是用初始的更快。

注意 java 的自定义比较函数。

## A.13 Leetcode 61

### Problem Description:

#### 旋转链表

给定一个链表，旋转链表，将链表每个节点向右移动 k 个位置，其中 k 是非负数。

### Sample:

input:

```
1
2 输入: 1->2->3->4->5->NULL, k = 2
3
4 输入: 0->1->2->NULL, k = 4
```

otput:

```
1
2 输出: 4->5->1->2->3->NULL
3 解释:
4 向右旋转 1 步: 5->1->2->3->4->NULL
5 向右旋转 2 步: 4->5->1->2->3->NULL
6
7 输出: 2->0->1->NULL
8 解释:
9 向右旋转 1 步: 2->0->1->NULL
10 向右旋转 2 步: 1->2->0->NULL
11 向右旋转 3 步: 0->1->2->NULL
12 向右旋转 4 步: 2->0->1->NULL
```

### Solution (Codes at B.13):

对 k 取模得到 mk，之后更改倒数第 mk 个 node 开始第子链表顺序到首位。

## A.14 Leetcode 121

### Problem Description:

#### 买卖股票的最佳时机



给定一个数组，它的第  $i$  个元素是一支给定股票第  $i$  天的价格。

如果你最多只允许完成一笔交易（即买入和卖出一支股票一次），设计一个算法来计算你能获取的最大利润。

注意：你不能在买入股票前卖出股票。

**Sample:**

input:

```
1
2 输入: [7,1,5,3,6,4]
3
4 输入: [7,6,4,3,1]
```

output:

```
1
2 输出: 5
3 解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大
      利润 = 6-1 = 5 。
4      注意利润不能是 7-1 = 6，因为卖出价格需要大于买入价格；同时，你不能在买入前卖出股
      票。
5
6 输出: 0
7 解释: 在这种情况下，没有交易完成，所以最大利润为 0。
```

**Solution** (Codes at B.14):

从后向前打表记录最大值，查询之后得出结果。

## A.15 Leetcode 138

**Problem Description:**

**复制带随机指针的链表**

给定一个链表，每个节点包含一个额外增加的随机指针，该指针可以指向链表中的任何节点或空节点。

要求返回这个链表的深拷贝。

我们用一个由  $n$  个节点组成的链表来表示输入/输出中的链表。每个节点用一个  $[val, random\_index]$  表示：

$val$ ：一个表示  $Node.val$  的整数。

$random\_index$ ：随机指针指向的节点索引（范围从 0 到  $n-1$ ）；如果不指向任何节点，则为  $null$ 。

$-10000 \leq Node.val \leq 10000$

$Node.random$  为空 ( $null$ ) 或指向链表中的节点。

节点数目不超过 1000。

**Sample:**

input:

```
1
2 输入: head = [[7,null],[13,0],[11,4],[10,2],[1,0]]
3
4 输入: head = [[1,1],[2,1]]
5 输入: head = [[3,null],[3,0],[3,null]]
6 输入: head = []
```

output:

```
1
2 输出: [[7,null],[13,0],[11,4],[10,2],[1,0]]
3 输出: [[1,1],[2,1]]
4 输出: [[3,null],[3,0],[3,null]]
5 输出: []
6 解释: 给定的链表为空（空指针），因此返回 null。
```

**Solution** (Codes at B.15):

map 存储节点 pair。

## A.16 Leetcode 141

### Problem Description:

#### 环形链表

给定一个链表，判断链表中是否有环。

为了表示给定链表中的环，我们使用整数 `pos` 来表示链表尾连接到链表中的位置（索引从 0 开始）。如果 `pos` 是 -1，则在该链表中没有环。

进阶：

你能用  $O(1)$ （即，常量）内存解决此问题吗？

### Sample:

input:

```
1
2 输入: head = [3,2,0,-4], pos = 1
3
4 输入: head = [1,2], pos = 0
5
6 输入: head = [1], pos = -1
```

output:

```
1
2 输出: true
3 解释: 链表中有一个环，其尾部连接到第二个节点。
4
5 输出: true
6 解释: 链表中有一个环，其尾部连接到第一个节点。
7
8 输出: false
9 解释: 链表中没有环。
```

**Solution** (Codes at B.16):

快慢指针循环跑圈相遇查重。

## A.17 Leetcode 202

### Problem Description:

#### 快乐数

编写一个算法来判断一个数 `n` 是不是快乐数。

快乐数定义为：对于一个正整数，每一次将该数替换为它每个位置上的数字的平方和，然后重复这个过程直到这个数变为 1，也可能是无限循环但始终变不到 1。如果可以变为 1，那么这个数就是快乐数。

如果 `n` 是快乐数就返回 `True`；不是，则返回 `False`。

### Sample:

input:

```
1
2 输入: 19
```

output:

```
1
2 输出: true
3 解释:
4 12 + 92 = 82
5 82 + 22 = 68
6 62 + 82 = 100
7 12 + 02 + 02 = 1
```

**Solution** (Codes at B.17):

快慢指针循环跑圈相遇查重。

## A.18 Leetcode 206

### Problem Description:

#### 反转链表

反转一个单链表。

#### Sample:

input:

```
1
2 输入: 1->2->3->4->5->NULL
```

output:

```
1
2 输出: 5->4->3->2->1->NULL
```

### Solution (Codes at B.18):

递归或先找到首尾节点之后 while 循环更新 next。

## A.19 Leetcode 209

### Problem Description:

#### 长度最小的子数组

给定一个含有  $n$  个正整数的数组和一个正整数  $s$ ，找出该数组中满足其和  $\geq s$  的长度最小的连续子数组，并返回其长度。如果不存在符合条件的子数组，返回 0。

进阶：

如果你已经完成了  $O(n)$  时间复杂度的解法，请尝试  $O(n \log n)$  时间复杂度的解法。

#### Sample:

input:

```
1
2 输入: s = 7, nums = [2,3,1,2,4,3]
```

output:

```
1
2 输出: 2
3 解释: 子数组 [4,3] 是该条件下的长度最小的子数组。
```

### Solution (Codes at B.19):

遍历一遍，滑动窗口更新数值。

或者前缀和，二分搜索数值。

## A.20 Leetcode 387

### Problem Description:

#### 字符串中的第一个唯一字符

给定一个字符串，找到它的第一个不重复的字符，并返回它的索引。如果不存在，则返回  $-1$ 。

#### Sample:

input:

```
1
2 leetcode
3 loveleetcode
```

output:

```
1
2 0
3 2
```

**Solution** (Codes at B.20):

找到字符出现的首位和末位进行判断，然后取最早出现的。

## A.21 Leetcode 876

### Problem Description:

#### 链表的中间结点

给定一个带有头结点 head 的非空单链表，返回链表的中间结点。

如果有两个中间结点，则返回第二个中间结点。

#### Sample:

input:

```
1
2 输入: [1,2,3,4,5]
3
4 输入: [1,2,3,4,5,6]
```

otput:

```
1
2 输出: 此列表中的结点 3 (序列化形式: [3,4,5])
3 返回的结点值为 3 。(测评系统对该结点序列化表述是 [3,4,5])。
4 注意, 我们返回了一个 ListNode 类型的对象 ans, 这样:
5 ans.val = 3, ans.next.val = 4, ans.next.next.val = 5, 以及 ans.next.next.next = NULL.
6
7 输出: 此列表中的结点 4 (序列化形式: [4,5,6])
8 由于该列表有两个中间结点, 值分别为 3 和 4, 我们返回第二个结点。
```

**Solution** (Codes at B.21):

快慢指针计数。

## 附录 B Code List

### B.1 Leetcode 1

C++

```
1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <queue>
7  #include <stack>
8  #include <string>
9  #include <vector>
10
11 using namespace std;
12
13 class Solution {
14 public:
15     vector<int> twoSum(vector<int>& nums, int target) {
16         vector<int> ans;
17         for (int i = 0; i < nums.size(); i++)
18             for (int j = i + 1; j < nums.size(); j++)
19                 if (nums[i] + nums[j] == target) {
20                     // u = i;
21                     // v = j;
22                     ans.push_back(i);
23                     ans.push_back(j);
24                     break;
25                 }
26         return ans;
```

```

27     }
28     void input(void) {
29         while (~scanf("%d %d", &n, &m))
30             for (int i = 0; i < n; i++) {
31                 scanf("%d", &t);
32                 numbers.push_back(t);
33             }
34     }
35     void solve(void) { twoSum(numbers, m); }
36     void output(void) { printf("%d %d\n", u, v); }
37
38     private:
39     int n, m, t;
40     int u, v;
41     vector<int> numbers;
42 };
43
44 int main() {
45     freopen("./assets/fipt.txt", "r", stdin);
46     freopen("./assets/fopt.txt", "w", stdout);
47
48     Solution sol;
49
50     sol.input();
51     sol.solve();
52     sol.output();
53
54     return 0;
55 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public int[] twoSum(int[] nums, int target) {
9          int[] ans = new int[2];
10         for (int i = 0; i < nums.length; i++)
11             for (int j = i + 1; j < nums.length; j++)
12                 if (nums[i] + nums[j] == target) {
13                     u = i;
14                     v = j;
15                     ans[0] = i;
16                     ans[1] = j;
17                     break;
18                 }
19         return ans;
20     }
21
22     public static void main(String[] args) throws FileNotFoundException {
23         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
24         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
25
26         System.setIn(fin);
27         System.setOut(fot);
28
29         Solution sol = new Solution();
30
31         sol.input();
32         sol.solve();
33         sol.output();
34     }
35
36     public void input() {
37         Scanner in = new Scanner(System.in);

```

```

38     while (in.hasNext()) {
39         n = in.nextInt();
40         m = in.nextInt();
41         for (int i = 0; i < n; i++)
42             numbers[i] = in.nextInt();
43     }
44
45     in.close();
46 }
47
48 public void solve() {
49     twoSum(numbers, m);
50 }
51
52 public void output() {
53     System.out.println(u + " " + v);
54 }
55
56 private int n, m, t;
57 private int u, v;
58 private int[] numbers = new int[10000];
59 }

```

## B.2 Leetcode 2

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 typedef struct ListNode {
15     int val;
16     ListNode* next;
17     ListNode(int x) : val(x), next(NULL) {}
18 } ListNode;
19
20 class Solution {
21 public:
22     ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
23         ListNode* res = new ListNode(0);
24         ListNode* u = l1;
25         ListNode* v = l2;
26         ListNode* cur = NULL;
27         int t = 0;
28
29         while (u != NULL || v != NULL || t != 0) {
30             if (cur == NULL) {
31                 cur = res;
32             } else {
33                 cur->next = new ListNode(0);
34                 cur = cur->next;
35             }
36             cur->next = NULL;
37
38             cur->val = t;
39             if (u != NULL) {
40                 cur->val += u->val;

```

```

41     u = u->next;
42 }
43 if (v != NULL) {
44     cur->val += v->val;
45     v = v->next;
46 }
47 t = cur->val / 10;
48 cur->val %= 10;
49 }
50
51 return res;
52 }
53 void input(void) {
54     while (~scanf("%d %d", &n, &m)) {
55         a = new ListNode(0);
56         b = new ListNode(0);
57         ListNode* u = a;
58         ListNode* v = b;
59
60         for (int i = 0; i < n; i++) {
61             u->next = new ListNode(0);
62             u = u->next;
63             u->next = NULL;
64
65             scanf("%d", &u->val);
66         }
67         for (int i = 0; i < m; i++) {
68             v->next = new ListNode(0);
69             v = v->next;
70             v->next = NULL;
71
72             scanf("%d", &v->val);
73         }
74
75         solve(a, b);
76     }
77 }
78 void show(ListNode* x) {
79     ListNode* cur = x;
80     while (cur != NULL) {
81         printf("%d", cur->val);
82         cur = cur->next;
83     }
84 }
85 void solve(ListNode* x, ListNode* y) {
86     ListNode* res;
87     res = addTwoNumbers(x, y);
88     output(res);
89 }
90 void output(ListNode* x) {
91     ListNode* cur = x->next;
92     while (cur != NULL) {
93         printf("%d", cur->val);
94         cur = cur->next;
95     }
96     cout << endl;
97 }
98
99 private:
100 int n, m;
101 ListNode *a, *b;
102 };
103
104 int main() {
105     freopen("./assets/fipt.txt", "r", stdin);
106     freopen("./assets/fopt.txt", "w", stdout);
107
108     Solution sol;
109

```

```

110     sol.input();
111
112     return 0;
113 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public class ListNode {
9          int val;
10         ListNode next;
11
12         ListNode(int x) {
13             val = x;
14         }
15     }
16
17     public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
18         ListNode res = new ListNode(0);
19         ListNode u = l1;
20         ListNode v = l2;
21         ListNode cur = null;
22         int t = 0;
23
24         while (u != null || v != null || t != 0) {
25             if (cur == null) {
26                 cur = res;
27             } else {
28                 cur.next = new ListNode(0);
29                 cur = cur.next;
30             }
31             cur.next = null;
32
33             cur.val = t;
34             if (u != null) {
35                 cur.val += u.val;
36                 u = u.next;
37             }
38             if (v != null) {
39                 cur.val += v.val;
40                 v = v.next;
41             }
42             t = cur.val / 10;
43             cur.val %= 10;
44         }
45
46         return res;
47     }
48
49     public static void main(String[] args) throws FileNotFoundException {
50         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
51         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
52
53         System.setIn(fin);
54         System.setOut(fot);
55
56         Solution sol = new Solution();
57
58         sol.input();
59     }
60
61     public void input() {
62         Scanner in = new Scanner(System.in);

```



```

63     while (in.hasNext()) {
64         n = in.nextInt();
65         m = in.nextInt();
66
67         ListNode a = new ListNode(0);
68         ListNode b = new ListNode(0);
69         ListNode u = a;
70         ListNode v = b;
71
72         for (int i = 0; i < n; i++) {
73             if (i != 0) {
74                 u.next = new ListNode(0);
75                 u = u.next;
76             }
77             u.val = in.nextInt();
78         }
79         for (int i = 0; i < m; i++) {
80             if (i != 0) {
81                 v.next = new ListNode(0);
82                 v = v.next;
83             }
84             v.val = in.nextInt();
85         }
86
87         solve(a, b);
88     }
89
90     in.close();
91 }
92
93 public void solve(ListNode x, ListNode y) {
94     ListNode res;
95     res = addTwoNumbers(x, y);
96     output(res);
97 }
98
99 public void output(ListNode x) {
100     while (x != null) {
101         System.out.print(x.val);
102         x = x.next;
103     }
104     System.out.println("");
105 }
106
107 private int n, m;
108 }

```

## B.3 Leetcode 3

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int lengthOfLongestSubstring(string s) {

```

```

17     int res = 0;
18     int sz = s.length();
19     int l = 0, r = 0;
20     if (sz == 0) return 0;
21     int mp[256];
22     memset(mp, -1, 256 * sizeof(int));
23
24     for (int i = 0; i < sz; i++)
25         if (mp[s[i]] == -1) {
26             r = i;
27             mp[s[i]] = i;
28         } else {
29             if (mp[s[i]] == -2) {
30                 r = i;
31                 mp[s[i]] = i;
32             } else {
33                 res = max(res, r - l + 1);
34                 int newL = mp[s[i]] + 1;
35                 for (int j = l; j < mp[s[i]] + 1; j++) mp[s[j]] = -2;
36                 l = newL;
37                 r = i;
38                 mp[s[i]] = i;
39             }
40         }
41
42     res = max(res, r - l + 1);
43     return res;
44 }
45 void input(void) {
46     while (cin >> a) {
47         solve(a);
48     }
49 }
50 void solve(string x) {
51     int res;
52     res = lengthOfLongestSubstring(x);
53     output(res);
54 }
55 void output(int x) { printf("%d\n", x); }
56
57 private:
58     string a;
59 };
60
61 int main() {
62     freopen("./assets/fipt.txt", "r", stdin);
63     freopen("./assets/fopt.txt", "w", stdout);
64
65     Solution sol;
66
67     sol.input();
68
69     return 0;
70 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Arrays;
6  import java.util.Scanner;
7
8  class Solution {
9      public int lengthOfLongestSubstring(String s) {
10         int res = 0;
11         int sz = s.length();
12         int l = 0, r = 0;

```

```

13     if (sz == 0)
14         return 0;
15     int[] mp = new int[256];
16     Arrays.fill(mp, -1);
17
18     for (int i = 0; i < sz; i++)
19         if (mp[s.charAt(i)] == -1) {
20             r = i;
21             mp[s.charAt(i)] = i;
22         } else {
23             if (mp[s.charAt(i)] == -2) {
24                 r = i;
25                 mp[s.charAt(i)] = i;
26             } else {
27                 res = Math.max(res, r - l + 1);
28                 int newL = mp[s.charAt(i)] + 1;
29                 for (int j = l; j < mp[s.charAt(i)] + 1; j++)
30                     mp[s.charAt(j)] = -2;
31                 l = newL;
32                 r = i;
33                 mp[s.charAt(i)] = i;
34             }
35         }
36
37     res = Math.max(res, r - l + 1);
38     return res;
39 }
40
41 public static void main(String[] args) throws FileNotFoundException {
42     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
43     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
44
45     System.setIn(fin);
46     System.setOut(fot);
47
48     Solution sol = new Solution();
49
50     sol.input();
51 }
52
53 public void input() {
54     Scanner in = new Scanner(System.in);
55     while (in.hasNext()) {
56         solve(in.nextLine());
57     }
58
59     in.close();
60 }
61
62 public void solve(String x) {
63     int res;
64     res = lengthOfLongestSubstring(x);
65     output(res);
66 }
67
68
69 public void output(int x) {
70     System.out.println(x);
71 }
72
73 private String a;
74 }

```

## B.4 Leetcode 6

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     string convert(string s, int numRows) {
17         if (s.length() <= 1) return s;
18         if (numRows == 1) return s;
19         string res = "";
20         vector<int> v;
21         int addItem = 2 * (numRows - 1);
22         int sz = s.length();
23         int item = 0;
24         int len;
25         int l, r;
26         while (item < sz) {
27             v.push_back(item);
28             res += s[item];
29             item += addItem;
30         }
31         v.push_back(item);
32         for (len = 1; len < numRows - 1; len++) {
33             for (int i = 0; i < v.size(); i++) {
34                 l = v[i] - len;
35                 r = v[i] + len;
36                 if (0 <= l && l < sz) {
37                     res += s[l];
38                 }
39                 if (0 <= r && r < sz) {
40                     res += s[r];
41                 }
42             }
43         }
44         for (int i = 0; i < v.size(); i++) {
45             r = v[i] + numRows - 1;
46             if (r < sz) {
47                 res += s[r];
48             }
49         }
50         return res;
51     }
52     void input(void) {
53         while (~scanf("%d", &n)) {
54             cin >> a;
55
56             solve(a, n);
57         }
58     }
59
60     void solve(string x, int y) {
61         string res;
62         res = convert(x, y);
63         output(res);
64     }
65     void output(string x) { cout << x << endl; }
66
67 private:
68     int n;
69     string a;

```

```

70 };
71
72 int main() {
73     freopen("./assets/fipt.txt", "r", stdin);
74     freopen("./assets/fopt.txt", "w", stdout);
75
76     Solution sol;
77
78     sol.input();
79
80     return 0;
81 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.ArrayList;
6  import java.util.Scanner;
7
8  class Solution {
9      public String convert(String s, int numRows) {
10         if (s.length() <= 1)
11             return s;
12         if (numRows == 1)
13             return s;
14         String res = "";
15         ArrayList<Integer> v = new ArrayList<Integer>();
16         int addItem = 2 * (numRows - 1);
17         int sz = s.length();
18         int item = 0;
19         int len;
20         int l, r;
21         while (item < sz) {
22             v.add(item);
23
24             res += s.charAt(item);
25             item += addItem;
26         }
27         v.add(item);
28         for (len = 1; len < numRows - 1; len++) {
29             for (int i = 0; i < v.size(); i++) {
30                 l = v.get(i) - len;
31                 r = v.get(i) + len;
32                 if (0 <= l && l < sz) {
33                     res += s.charAt(l);
34                 }
35                 if (0 <= r && r < sz) {
36                     res += s.charAt(r);
37                 }
38             }
39         }
40         for (int i = 0; i < v.size(); i++) {
41             r = v.get(i) + numRows - 1;
42             if (r < sz) {
43                 res += s.charAt(r);
44             }
45         }
46         return res;
47     }
48
49     public static void main(String[] args) throws FileNotFoundException {
50         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
51         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
52
53         System.setIn(fin);
54         System.setOut(fot);

```

```

55
56     Solution sol = new Solution();
57
58     sol.input();
59 }
60
61 public void input() {
62     Scanner in = new Scanner(System.in);
63     while (in.hasNext()) {
64         n = in.nextInt();
65
66         a = in.next();
67
68         solve(a, n);
69     }
70
71     in.close();
72 }
73
74 public void solve(String x, int y) {
75     String res;
76     res = convert(x, y);
77     output(res);
78
79 }
80
81 public void output(String x) {
82     System.out.println(x);
83 }
84
85 private int n;
86 private String a;
87 }

```

## B.5 Leetcode 11

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int maxArea(vector<int>& height) {
17         int res = 0;
18         int sz = height.size();
19         int l = 0, r = sz - 1;
20         while (l < r) {
21             res = max(res, min(height[l], height[r]) * (r - l));
22             if (height[l] < height[r])
23                 l++;
24             else
25                 r--;
26         }
27
28         return res;
29     }

```

```

30
31 void input(void) {
32     while (~scanf("%d", &n)) {
33         int t;
34         for (int i = 0; i < n; i++) {
35             scanf("%d", &t);
36             a.push_back(t);
37         }
38         solve(a);
39     }
40 }
41 void solve(vector<int>& x) {
42     int res;
43     res = maxArea(x);
44     output(res);
45 }
46 void output(int x) { printf("%d\n", x); }
47
48 private:
49     int n;
50     vector<int> a;
51 };
52
53 int main() {
54     freopen("./assets/fipt.txt", "r", stdin);
55     freopen("./assets/fopt.txt", "w", stdout);
56
57     Solution sol;
58
59     sol.input();
60
61     return 0;
62 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public int maxArea(int[] height) {
9          int res = 0;
10         int sz = height.length;
11         int l = 0, r = sz - 1;
12         while (l < r) {
13             res = Math.max(res, Math.min(height[l], height[r]) * (r - l));
14             if (height[l] < height[r])
15                 l++;
16             else
17                 r--;
18         }
19
20         return res;
21     }
22
23     public static void main(String[] args) throws FileNotFoundException {
24         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
25         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
26
27         System.setIn(fin);
28         System.setOut(fot);
29
30         Solution sol = new Solution();
31
32         sol.input();
33     }

```

```

34
35 public void input() {
36     Scanner in = new Scanner(System.in);
37     while (in.hasNext()) {
38         n = in.nextInt();
39         a = new int[n];
40         for (int i = 0; i < n; i++)
41             a[i] = in.nextInt();
42         solve(a);
43     }
44
45     in.close();
46 }
47
48 public void solve(int[] x) {
49     int res;
50     res = maxArea(x);
51     output(res);
52 }
53
54
55 public void output(int x) {
56     System.out.println(x);
57 }
58
59 private int n;
60 private int[] a;
61 }

```

## B.6 Leetcode 15

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     vector<vector<int>> > threeSum(vector<int>& nums) {
17         vector<vector<int>> > res;
18         int sz = nums.size();
19         int i, j, k;
20         int r;
21         if (sz < 3) return res;
22         sort(nums.begin(), nums.end());
23
24         for (i = 0; i < sz; i++) {
25             if (nums[i] > 0) break;
26             if (i > 0 && nums[i] == nums[i - 1]) continue;
27             j = i + 1;
28             k = sz - 1;
29             while (j < k) {
30                 int tmp = nums[i] + nums[j] + nums[k];
31                 if (tmp < 0) {
32                     j++;
33                 } else if (tmp > 0) {
34                     k--;

```



```

35     } else {
36         res.push_back({nums[i], nums[j], nums[k]});
37         j++;
38         k--;
39         while (j < k && nums[j] == nums[j - 1]) j++;
40         while (j < k && nums[k] == nums[k + 1]) k--;
41     }
42 }
43 }
44
45 return res;
46 }
47
48 void input(void) {
49     while (~scanf("%d", &n)) {
50         int t;
51         for (int i = 0; i < n; i++) {
52             scanf("%d", &t);
53             a.push_back(t);
54         }
55         solve(a);
56         a.clear();
57     }
58 }
59 void solve(vector<int>& x) {
60     vector<vector<int>> > res;
61     res = threeSum(x);
62     output(res);
63 }
64 void output(vector<vector<int>> > x) {
65     for (int i = 0; i < x.size(); i++)
66         printf("%d %d %d\n", x[i][0], x[i][1], x[i][2]);
67     puts("");
68 }
69
70 private:
71     int n;
72     vector<int> a;
73 };
74
75 int main() {
76     freopen("./assets/fipt.txt", "r", stdin);
77     freopen("./assets/fopt.txt", "w", stdout);
78
79     Solution sol;
80
81     sol.input();
82
83     return 0;
84 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.ArrayList;
6  import java.util.Arrays;
7  import java.util.List;
8  import java.util.Scanner;
9
10 class Solution {
11     public List<List<Integer>> threeSum(int[] nums) {
12         List<List<Integer>> res = new ArrayList<>();
13         Arrays.sort(nums);
14         int sz = nums.length;
15         for (int i = 0; i < sz; i++) {
16             if (nums[i] > 0) {

```

```

17         break;
18     }
19     if (i > 0 && nums[i - 1] == nums[i]) {
20         continue;
21     }
22     int j = i + 1, k = sz - 1;
23     while (j < k) {
24         int t = nums[i] + nums[j] + nums[k];
25         if (t > 0) {
26             k--;
27         } else if (t < 0) {
28             j++;
29         } else {
30             res.add(Arrays.asList(nums[i], nums[j], nums[k]));
31             j++;
32             k--;
33             while (j < k && nums[j - 1] == nums[j]) {
34                 j++;
35             }
36             while (j < k && nums[k] == nums[k + 1]) {
37                 k--;
38             }
39         }
40     }
41 }
42 return res;
43 }
44
45 public static void main(String[] args) throws FileNotFoundException {
46     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
47     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
48
49     System.setIn(fin);
50     System.setOut(fot);
51
52     Solution sol = new Solution();
53
54     sol.input();
55 }
56
57 public void input() {
58     Scanner in = new Scanner(System.in);
59     while (in.hasNext()) {
60         n = in.nextInt();
61         a = new int[n];
62         for (int i = 0; i < n; i++)
63             a[i] = in.nextInt();
64         solve(a);
65     }
66
67     in.close();
68 }
69
70 public void solve(int[] x) {
71     List<List<Integer>> res;
72     res = threeSum(x);
73     output(res);
74 }
75
76
77 public void output(List<List<Integer>> x) {
78     System.out.println(x);
79 }
80
81 private int n;
82 private int[] a;
83 }

```

## B.7 Leetcode 16

C++

```
1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int threeSumClosest(vector<int>& nums, int target) {
17         int res = nums[0] + nums[1] + nums[2];
18         sort(nums.begin(), nums.end());
19         int sz = nums.size();
20         for (int i = 0; i < sz; i++) {
21             if (i != 0 && nums[i - 1] == nums[i]) {
22                 continue;
23             }
24             int j = i + 1, k = sz - 1;
25             while (j < k) {
26                 int t = nums[i] + nums[j] + nums[k];
27                 if (t < target) {
28                     if (abs(target - t) < abs(target - res)) res = t;
29                     j++;
30                     while (j < k && nums[j - 1] == nums[j]) j++;
31                 } else if (t > target) {
32                     if (abs(target - t) < abs(target - res)) res = t;
33                     k--;
34                     while (j < k && nums[k] == nums[k + 1]) k--;
35                 } else {
36                     return target;
37                 }
38             }
39         }
40         return res;
41     }
42
43     void input(void) {
44         while (~scanf("%d %d", &n, &m)) {
45             int t;
46             for (int i = 0; i < n; i++) {
47                 scanf("%d", &t);
48                 a.push_back(t);
49             }
50             solve(a, m);
51             a.clear();
52         }
53     }
54     void solve(vector<int>& x, int y) {
55         int res;
56         res = threeSumClosest(x, y);
57         output(res);
58     }
59     void output(int x) { printf("%d\n", x); }
60
61 private:
62     int n, m;
63     vector<int> a;
64 };
65
```

```

66 int main() {
67     freopen("./assets/fipt.txt", "r", stdin);
68     freopen("./assets/fopt.txt", "w", stdout);
69
70     Solution sol;
71
72     sol.input();
73
74     return 0;
75 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Arrays;
6  import java.util.Scanner;
7
8  class Solution {
9      public int threeSumClosest(int[] nums, int target) {
10         int res = nums[0] + nums[1] + nums[2];
11         Arrays.sort(nums);
12         int sz = nums.length;
13         for (int i = 0; i < sz; i++) {
14             if (i != 0 && nums[i - 1] == nums[i]) {
15                 continue;
16             }
17             int j = i + 1, k = sz - 1;
18             while (j < k) {
19                 int t = nums[i] + nums[j] + nums[k];
20                 if (t < target) {
21                     if (Math.abs(target - t) < Math.abs(target - res))
22                         res = t;
23                     j++;
24                     while (j < k && nums[j - 1] == nums[j])
25                         j++;
26                 } else if (t > target) {
27                     if (Math.abs(target - t) < Math.abs(target - res))
28                         res = t;
29                     k--;
30                     while (j < k && nums[k] == nums[k + 1])
31                         k--;
32                 } else {
33                     return target;
34                 }
35             }
36         }
37         return res;
38     }
39
40     public static void main(String[] args) throws FileNotFoundException {
41         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
42         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
43
44         System.setIn(fin);
45         System.setOut(fot);
46
47         Solution sol = new Solution();
48
49         sol.input();
50     }
51
52     public void input() {
53         Scanner in = new Scanner(System.in);
54         while (in.hasNext()) {
55             n = in.nextInt();
56             a = new int[n];

```

```

57     for (int i = 0; i < n; i++)
58         a[i] = in.nextInt();
59     solve(a);
60 }
61
62     in.close();
63 }
64
65     public void solve(int[] x, int y) {
66         int res;
67         res = threeSumClosest(x, y);
68         output(res);
69     }
70
71     public void output(int x) {
72         System.out.println(x);
73     }
74
75     private int n;
76     private int[] a;
77 }
78

```

## B.8 Leetcode 19

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 typedef struct ListNode {
15     int val;
16     ListNode* next;
17     ListNode(int x) : val(x), next(NULL) {}
18 } ListNode;
19
20 class Solution {
21 public:
22     int deleteNode(ListNode* x, int target) {
23         if (x == NULL) return 0;
24
25         int u = deleteNode(x->next, target);
26
27         if (u != -1) {
28             if (u == target) {
29                 ListNode* y = x->next->next;
30                 delete x->next;
31                 x->next = y;
32             } else {
33                 return u + 1;
34             }
35         }
36
37         return -1;
38     }
39     ListNode* removeNthFromEnd(ListNode* head, int n) {
40         int u = deleteNode(head, n);
41

```

```

41     if (u != -1) {
42         ListNode* t = head;
43         head = head->next;
44         delete t;
45     }
46
47     return head;
48 }
49
50 void input(void) {
51     while (~scanf("%d %d", &n, &m)) {
52         a = new ListNode(0);
53         ListNode* u = a;
54
55         for (int i = 0; i < m; i++) {
56             if (i != 0) {
57                 u->next = new ListNode(0);
58                 u = u->next;
59             }
60
61             scanf("%d", &u->val);
62         }
63
64         solve(a, n);
65     }
66 }
67 void solve(ListNode* x, int y) {
68     ListNode* res;
69     res = removeNthFromEnd(x, y);
70     output(res);
71 }
72 void output(ListNode* x) {
73     ListNode* cur = x;
74     while (cur != NULL) {
75         printf("%d", cur->val);
76         cur = cur->next;
77     }
78 }
79
80 private:
81     int n, m;
82     ListNode* a;
83 };
84
85 int main() {
86     freopen("./assets/fipt.txt", "r", stdin);
87     freopen("./assets/fopt.txt", "w", stdout);
88
89     Solution sol;
90
91     sol.input();
92
93     return 0;
94 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public class ListNode {
9          int val;
10         ListNode next;
11
12         ListNode(int x) {

```

```

13     val = x;
14 }
15 }
16
17 public int deleteNode(ListNode x, int target) {
18     if (x == null)
19         return 0;
20
21     int u = deleteNode(x.next, target);
22
23     if (u != -1) {
24         if (u == target) {
25             ListNode y = x.next.next;
26             x.next = y;
27         } else {
28             return u + 1;
29         }
30     }
31
32     return -1;
33 }
34
35 public ListNode removeNthFromEnd(ListNode head, int n) {
36     int u = deleteNode(head, n);
37     if (u != -1) {
38         head = head.next;
39     }
40     return head;
41 }
42
43 public static void main(String[] args) throws FileNotFoundException {
44     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
45     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
46
47     System.setIn(fin);
48     System.setOut(fot);
49
50     Solution sol = new Solution();
51
52     sol.input();
53 }
54
55 public void input() {
56     Scanner in = new Scanner(System.in);
57     while (in.hasNext()) {
58         n = in.nextInt();
59         m = in.nextInt();
60
61         ListNode a = new ListNode(0);
62         ListNode u = a;
63
64         for (int i = 0; i < m; i++) {
65             if (i != 0) {
66                 u.next = new ListNode(0);
67                 u = u.next;
68             }
69             u.val = in.nextInt();
70         }
71
72         solve(a, n);
73     }
74
75     in.close();
76 }
77
78 public void solve(ListNode x, int y) {
79     ListNode res;
80     res = removeNthFromEnd(x, y);
81     output(res);

```

```

82     }
83
84     public void output(ListNode x) {
85         while (x != null) {
86             System.out.print(x.val);
87             x = x.next;
88         }
89         System.out.println("");
90     }
91
92     private int n, m;
93 }

```

## B.9 Leetcode 25

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 typedef struct ListNode {
15     int val;
16     ListNode* next;
17     ListNode(int x) : val(x), next(NULL) {}
18 } ListNode;
19
20 class Solution {
21 public:
22     void reverseGroup(ListNode* u, ListNode* v) {
23         if (u != v) {
24             if (u->next == v) {
25                 v->next = u;
26             } else {
27                 ListNode* t = u->next;
28                 reverseGroup(u->next, v);
29                 t->next = u;
30             }
31         }
32     }
33     ListNode* reverseKGroup(ListNode* head, int k) {
34         if (k == 1) return head;
35
36         ListNode* fakeHead = new ListNode(0);
37         fakeHead->next = head;
38         ListNode* pre = fakeHead;
39         ListNode* cur = head;
40         ListNode* u = NULL;
41         ListNode* t = NULL;
42         ListNode* v = NULL;
43         int cnt = 0;
44         while (cur != NULL) {
45             cnt++;
46             if (cnt == k) {
47                 u = pre->next;
48                 v = cur;
49                 t = cur->next;
50                 reverseGroup(u, v);

```



```

51     pre->next = v;
52     u->next = t;
53     pre = u;
54     cur = u;
55     cnt = 0;
56 }
57 cur = cur->next;
58 }
59 return fakeHead->next;
60 }
61 void input(void) {
62     while (~scanf("%d %d", &n, &m)) {
63         a = new ListNode(0);
64         ListNode* u = a;
65
66         for (int i = 0; i < m; i++) {
67             if (i != 0) {
68                 u->next = new ListNode(0);
69                 u = u->next;
70             }
71
72             scanf("%d", &u->val);
73         }
74
75         solve(a, n);
76     }
77 }
78 void solve(ListNode* x, int y) {
79     ListNode* res;
80     res = reverseKGroup(x, y);
81     output(res);
82 }
83 void output(ListNode* x) {
84     ListNode* cur = x;
85     while (cur != NULL) {
86         printf("%d", cur->val);
87         cur = cur->next;
88     }
89     cout << endl;
90 }
91
92 private:
93     int n, m;
94     ListNode* a;
95 };
96
97 int main() {
98     freopen("./assets/fipt.txt", "r", stdin);
99     freopen("./assets/fopt.txt", "w", stdout);
100
101     Solution sol;
102
103     sol.input();
104
105     return 0;
106 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public class ListNode {
9          int val;
10         ListNode next;

```

```

11     ListNode(int x) {
12         val = x;
13     }
14 }
15
16
17 void reverseGroup(ListNode u, ListNode v) {
18     if (u != v) {
19         if (u.next == v) {
20             v.next = u;
21         } else {
22             ListNode t = u.next;
23             reverseGroup(u.next, v);
24             t.next = u;
25         }
26     }
27 }
28
29 public ListNode reverseKGroup(ListNode head, int k) {
30     if (k == 1)
31         return head;
32     ListNode fakeHead = new ListNode(0);
33     fakeHead.next = head;
34     ListNode pre = fakeHead;
35     ListNode cur = head;
36     ListNode u = null;
37     ListNode t = null;
38     ListNode v = null;
39     int cnt = 0;
40     while (cur != null) {
41         cnt++;
42         if (cnt == k) {
43             u = pre.next;
44             v = cur;
45             t = cur.next;
46             reverseGroup(u, v);
47             pre.next = v;
48             u.next = t;
49             pre = u;
50             cur = u;
51             cnt = 0;
52         }
53         cur = cur.next;
54     }
55     return fakeHead.next;
56 }
57
58 public static void main(String[] args) throws FileNotFoundException {
59     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
60     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
61
62     System.setIn(fin);
63     System.setOut(fot);
64
65     Solution sol = new Solution();
66
67     sol.input();
68 }
69
70 public void input() {
71     Scanner in = new Scanner(System.in);
72     while (in.hasNext()) {
73         n = in.nextInt();
74         m = in.nextInt();
75
76         ListNode a = new ListNode(0);
77         ListNode u = a;
78
79         for (int i = 0; i < m; i++) {

```

```

80         if (i != 0) {
81             u.next = new ListNode(0);
82             u = u.next;
83         }
84         u.val = in.nextInt();
85     }
86
87     solve(a, n);
88 }
89
90 in.close();
91 }
92
93 public void solve(ListNode x, int y) {
94     ListNode res;
95     res = reverseKGroup(x, y);
96     output(res);
97 }
98
99 public void output(ListNode x) {
100     while (x != null) {
101         System.out.print(x.val);
102         x = x.next;
103     }
104     System.out.println("");
105 }
106
107 private int n, m;
108 }

```

## B.10 Leetcode 26

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int removeDuplicates(vector<int>& nums) {
17         int res = 0;
18         int sz = nums.size();
19         if (sz == 0) return 0;
20         int cur = 0;
21         for (int i = 0; i < sz; i++)
22             if (nums[cur] != nums[i]) {
23                 cur++;
24                 nums[cur] = nums[i];
25             }
26         res = cur + 1;
27         return res;
28     }
29
30     void input(void) {
31         while (~scanf("%d", &n)) {
32             int t;
33             for (int i = 0; i < n; i++) {

```

```

34     scanf("%d", &t);
35     a.push_back(t);
36 }
37 solve(a);
38 a.clear();
39 }
40 }
41 void solve(vector<int>& x) {
42     int res;
43     res = removeDuplicates(x);
44     output(res);
45 }
46 void output(int x) { printf("%d\n", x); }
47
48 private:
49     int n;
50     vector<int> a;
51 };
52
53 int main() {
54     freopen("./assets/fipt.txt", "r", stdin);
55     freopen("./assets/fopt.txt", "w", stdout);
56
57     Solution sol;
58
59     sol.input();
60
61     return 0;
62 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public int removeDuplicates(int[] nums) {
9          int res = 0;
10         int sz = nums.length;
11         if (sz == 0)
12             return 0;
13         int cur = 0;
14         for (int i = 0; i < sz; i++)
15             if (nums[cur] != nums[i]) {
16                 cur++;
17                 nums[cur] = nums[i];
18             }
19         res = cur + 1;
20         return res;
21     }
22
23     public static void main(String[] args) throws FileNotFoundException {
24         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
25         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
26
27         System.setIn(fin);
28         System.setOut(fot);
29
30         Solution sol = new Solution();
31
32         sol.input();
33     }
34
35     public void input() {
36         Scanner in = new Scanner(System.in);
37         while (in.hasNext()) {

```

```

38     n = in.nextInt();
39     a = new int[n];
40     for (int i = 0; i < n; i++)
41         a[i] = in.nextInt();
42     solve(a);
43 }
44
45     in.close();
46 }
47
48     public void solve(int[] x) {
49         int res;
50         res = removeDuplicates(x);
51         output(res);
52     }
53
54     public void output(int x) {
55         System.out.println(x);
56     }
57 }
58
59     private int n;
60     private int[] a;
61 }

```

## B.11 Leetcode 42

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int trap(vector<int>& height) {
17         int res = 0;
18         int sz = height.size();
19         if (sz == 0) return 0;
20
21         vector<int> lmx(sz, 0), rmx(sz, 0);
22
23         for (int i = 1, j = sz - 2; i < sz; i++, j--) {
24             lmx[i] = max(lmx[i - 1], height[i - 1]);
25             rmx[j] = max(rmx[j + 1], height[j + 1]);
26         }
27         for (int i = 1; i < sz - 1; i++)
28             res += max(0, min(lmx[i], rmx[i]) - height[i]);
29         return res;
30     }
31
32     void input(void) {
33         while (~scanf("%d", &n)) {
34             int t;
35             for (int i = 0; i < n; i++) {
36                 scanf("%d", &t);
37                 a.push_back(t);
38             }

```

```

39     solve(a);
40     a.clear();
41 }
42 }
43 void solve(vector<int>& x) {
44     int res;
45     res = trap(x);
46     output(res);
47 }
48 void output(int x) { printf("%d\n", x); }
49
50 private:
51     int n;
52     vector<int> a;
53 };
54
55 int main() {
56     freopen("./assets/fipt.txt", "r", stdin);
57     freopen("./assets/fopt.txt", "w", stdout);
58
59     Solution sol;
60
61     sol.input();
62
63     return 0;
64 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public int trap(int[] height) {
9          int res = 0;
10         int sz = height.length;
11         if (sz == 0)
12             return 0;
13
14         int[] lmx = new int[sz];
15         int[] rmx = new int[sz];
16
17         for (int i = 1, j = sz - 2; i < sz; i++, j--) {
18             lmx[i] = Math.max(lmx[i - 1], height[i - 1]);
19             rmx[j] = Math.max(rmx[j + 1], height[j + 1]);
20         }
21         for (int i = 1; i < sz - 1; i++)
22             res += Math.max(0, Math.min(lmx[i], rmx[i]) - height[i]);
23         return res;
24     }
25
26     public static void main(String[] args) throws FileNotFoundException {
27         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
28         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
29
30         System.setIn(fin);
31         System.setOut(fot);
32
33         Solution sol = new Solution();
34
35         sol.input();
36     }
37
38     public void input() {
39         Scanner in = new Scanner(System.in);
40         while (in.hasNext()) {

```

```

41     n = in.nextInt();
42     a = new int[n];
43     for (int i = 0; i < n; i++)
44         a[i] = in.nextInt();
45     solve(a);
46 }
47
48     in.close();
49 }
50
51     public void solve(int[] x) {
52         int res;
53         res = trap(x);
54         output(res);
55     }
56
57     public void output(int x) {
58         System.out.println(x);
59     }
60 }
61
62     private int n;
63     private int[] a;
64 }

```

## B.12 Leetcode 56

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16
17     vector<vector<int> > merge(vector<vector<int> >& intervals) {
18         vector<vector<int> > res;
19         sort(intervals.begin(), intervals.end());
20
21         for (int i = 0; i < intervals.size(); i++) {
22             if (res.empty()) {
23                 res.push_back(intervals[i]);
24             } else {
25                 if (res.back()[1] < intervals[i][0]) {
26                     res.push_back(intervals[i]);
27                 } else {
28                     if (res.back()[1] < intervals[i][1]) res.back()[1] = intervals[i][1];
29                 }
30             }
31         }
32
33         return res;
34     }
35
36     void input(void) {
37         while (~scanf("%d", &n)) {
38             int u, v;
39             for (int i = 0; i < n; i++) {

```

```

39     vector<int> t;
40     scanf("%d %d", &u, &v);
41
42     t.push_back(u);
43     t.push_back(v);
44     a.push_back(t);
45 }
46
47 solve(a);
48 a.clear();
49 }
50 }
51 void solve(vector<vector<int> >& x) {
52     vector<vector<int> > res;
53     res = merge(x);
54     output(res);
55 }
56 void output(vector<vector<int> >& x) {
57     vector<vector<int> > cur = x;
58     for (int i = 0; i < cur.size(); i++)
59         printf("%d %d\n", cur[i][0], cur[i][1]);
60 }
61
62 private:
63     int n;
64     vector<vector<int> > a;
65 };
66
67 int main() {
68     freopen("./assets/fipt.txt", "r", stdin);
69     freopen("./assets/fopt.txt", "w", stdout);
70
71     Solution sol;
72
73     sol.input();
74
75     return 0;
76 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.ArrayList;
6  import java.util.Arrays;
7  import java.util.Comparator;
8  import java.util.Scanner;
9
10 class Solution {
11     public int[][] merge(int[][] intervals) {
12         ArrayList<int[]> res = new ArrayList<int[]>();
13         Arrays.sort(intervals, new Comparator<int[]>() {
14             @Override
15             public int compare(int[] l, int[] r) {
16                 return l[0] - r[0];
17             }
18         });
19         int sz = intervals.length;
20         for (int i = 0; i < sz; i++) {
21             int l = intervals[i][0];
22             int r = intervals[i][1];
23             while (i < sz - 1 && intervals[i + 1][0] <= r) {
24                 r = Math.max(r, intervals[i + 1][1]);
25                 i++;
26             }
27             res.add(new int[] { l, r });
28         }
29     }
30 }

```



```

29
30     return res.toArray(new int[res.size()][2]);
31 }
32
33 public static void main(String[] args) throws FileNotFoundException {
34     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
35     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
36
37     System.setIn(fin);
38     System.setOut(fot);
39
40     Solution sol = new Solution();
41
42     sol.input();
43 }
44
45 public void input() {
46     Scanner in = new Scanner(System.in);
47     while (in.hasNext()) {
48         n = in.nextInt();
49
50         vals = new int[n][2];
51
52         for (int i = 0; i < n; i++) {
53             vals[i][0] = in.nextInt();
54             vals[i][1] = in.nextInt();
55         }
56
57         solve(vals);
58     }
59
60     in.close();
61 }
62
63 public void solve(int[][] x) {
64     int[][] res;
65     res = merge(x);
66     output(res);
67 }
68
69 public void output(int[][] x) {
70     int sz = x.length;
71     for (int i = 0; i < sz; i++)
72         System.out.println(x[i][0] + " " + x[i][1]);
73 }
74
75 private int n;
76 private int[][] vals;
77
78 }

```

## B.13 Leetcode 61

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;

```

```

13
14 typedef struct ListNode {
15     int val;
16     ListNode* next;
17     ListNode(int x) : val(x), next(NULL) {}
18 } ListNode;
19
20 class Solution {
21 public:
22     ListNode* rotateRight(ListNode* head, int k) {
23         if (head == NULL) return head;
24         ListNode* res = head;
25         ListNode* cur = head;
26         ListNode* pre = NULL;
27         ListNode* st = NULL;
28         ListNode* en = NULL;
29         vector<ListNode*> v;
30         while (cur != NULL) {
31             v.push_back(cur);
32
33             if (cur->next == NULL) en = cur;
34             cur = cur->next;
35         }
36
37         int lenOfList = v.size();
38         int mk = k % lenOfList;
39         if (mk != 0) {
40             pre = v[lenOfList - mk - 1];
41             st = v[lenOfList - mk];
42             pre->next = NULL;
43             en->next = res;
44             res = st;
45         }
46
47         return res;
48     }
49
50     void input(void) {
51         while (~scanf("%d %d", &n, &m)) {
52             a = new ListNode(0);
53             ListNode* u = a;
54
55             for (int i = 0; i < m; i++) {
56                 if (i != 0) {
57                     u->next = new ListNode(0);
58                     u = u->next;
59                 }
60
61                 scanf("%d", &u->val);
62             }
63
64             solve(a, n);
65         }
66     }
67     void solve(ListNode* x, int y) {
68         ListNode* res;
69         res = rotateRight(x, y);
70         output(res);
71     }
72     void output(ListNode* x) {
73         ListNode* cur = x;
74         while (cur != NULL) {
75             printf("%d", cur->val);
76             cur = cur->next;
77         }
78         cout << endl;
79     }
80
81 private:

```

```

82     int n, m;
83     ListNode* a;
84 };
85
86 int main() {
87     freopen("./assets/fipt.txt", "r", stdin);
88     freopen("./assets/fopt.txt", "w", stdout);
89
90     Solution sol;
91
92     sol.input();
93
94     return 0;
95 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.LinkedList;
6  import java.util.Scanner;
7
8  class Solution {
9      public class ListNode {
10         int val;
11         ListNode next;
12
13         ListNode(int x) {
14             val = x;
15         }
16     }
17
18     public ListNode rotateRight(ListNode head, int k) {
19         if (head == null)
20             return head;
21         ListNode res = head;
22         ListNode cur = head;
23         ListNode pre = null;
24         ListNode st = null;
25         ListNode en = null;
26         LinkedList<ListNode> v = new LinkedList<ListNode>();
27         while (cur != null) {
28             v.add(cur);
29
30             if (cur.next == null)
31                 en = cur;
32             cur = cur.next;
33         }
34
35         int lenOfList = v.size();
36         int mk = k % lenOfList;
37         if (mk != 0) {
38             pre = v.get(lenOfList - mk - 1);
39             st = v.get(lenOfList - mk);
40             pre.next = null;
41             en.next = res;
42             res = st;
43         }
44
45         return res;
46     }
47
48     public static void main(String[] args) throws FileNotFoundException {
49         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
50         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
51
52         System.setIn(fin);

```

```

53     System.setOut(fot);
54
55     Solution sol = new Solution();
56
57     sol.input();
58 }
59
60 public void input() {
61     Scanner in = new Scanner(System.in);
62     while (in.hasNext()) {
63         n = in.nextInt();
64         m = in.nextInt();
65
66         ListNode a = new ListNode(0);
67         ListNode u = a;
68
69         for (int i = 0; i < m; i++) {
70             if (i != 0) {
71                 u.next = new ListNode(0);
72                 u = u.next;
73             }
74             u.val = in.nextInt();
75         }
76
77         solve(a, n);
78     }
79
80     in.close();
81 }
82
83 public void solve(ListNode x, int y) {
84     ListNode res;
85     res = rotateRight(x, y);
86     output(res);
87 }
88
89 public void output(ListNode x) {
90     while (x != null) {
91         System.out.print(x.val);
92         x = x.next;
93     }
94     System.out.println("");
95 }
96
97 private int n, m;
98 }

```

## B.14 Leetcode 121

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int maxProfit(vector<int>& prices) {

```

```

17     int res = 0;
18     int rmx = 0;
19     for (int i = prices.size() - 2; i >= 0; i--) {
20         rmx = max(rmx, prices[i + 1]);
21         res = max(res, max(0, (rmx - prices[i])));
22     }
23     return res;
24 }
25
26 void input(void) {
27     while (~scanf("%d", &n)) {
28         int t;
29         for (int i = 0; i < n; i++) {
30             scanf("%d", &t);
31             a.push_back(t);
32         }
33         solve(a);
34         a.clear();
35     }
36 }
37 void solve(vector<int>& x) {
38     int res;
39     res = maxProfit(x);
40     output(res);
41 }
42 void output(int x) { printf("%d\n", x); }
43
44 private:
45     int n;
46     vector<int> a;
47 };
48
49 int main() {
50     freopen("./assets/fipt.txt", "r", stdin);
51     freopen("./assets/fopt.txt", "w", stdout);
52
53     Solution sol;
54
55     sol.input();
56
57     return 0;
58 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public int maxProfit(int[] prices) {
9          int res = 0;
10         int rmx = 0;
11         for (int i = prices.length - 2; i >= 0; i--) {
12             rmx = Math.max(rmx, prices[i + 1]);
13             res = Math.max(res, Math.max(0, (rmx - prices[i])));
14         }
15         return res;
16     }
17
18     public static void main(String[] args) throws FileNotFoundException {
19         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
20         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
21
22         System.setIn(fin);
23         System.setOut(fot);
24

```

```

25     Solution sol = new Solution();
26
27     sol.input();
28 }
29
30 public void input() {
31     Scanner in = new Scanner(System.in);
32     while (in.hasNext()) {
33         n = in.nextInt();
34         a = new int[n];
35         for (int i = 0; i < n; i++)
36             a[i] = in.nextInt();
37         solve(a);
38     }
39
40     in.close();
41 }
42
43 public void solve(int[] x) {
44     int res;
45     res = maxProfit(x);
46     output(res);
47 }
48
49
50 public void output(int x) {
51     System.out.println(x);
52 }
53
54 private int n;
55 private int[] a;
56 }

```

## B.15 Leetcode 138

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Node {
15 public:
16     int val;
17     Node* next;
18     Node* random;
19
20     Node(int _val) {
21         val = _val;
22         next = NULL;
23         random = NULL;
24     }
25 };
26 class Solution {
27 public:
28     Node* copyRandomList(Node* head) {
29         if (head == NULL) return NULL;
30

```

```

31 Node* res = new Node(head->val);
32 Node* cur_h = head;
33 Node* cur_r = res;
34 map<Node*, Node*> mp;
35 mp.insert(pair<Node*, Node*>(NULL, NULL));
36
37 while (cur_h != NULL) {
38     if (cur_h != head) {
39         cur_r->next = new Node(cur_h->val);
40         cur_r = cur_r->next;
41     }
42     mp.insert(pair<Node*, Node*>(cur_h, cur_r));
43     cur_h = cur_h->next;
44 }
45
46 cur_h = head;
47 cur_r = res;
48 while (cur_h != NULL) {
49     cur_r->random = mp[cur_h->random];
50     cur_r = cur_r->next;
51
52     cur_h = cur_h->next;
53 }
54
55 return res;
56 }
57 void input(void) {
58     while (~scanf("%d", &n)) {
59         a = new Node(0);
60         Node* u = a;
61         int x;
62         vector<int> v;
63         vector<Node*> record;
64
65         for (int i = 0; i < n; i++) {
66             if (i != 0) {
67                 u->next = new Node(0);
68                 u = u->next;
69             }
70
71             scanf("%d %d", &u->val, &x);
72             v.push_back(x);
73             record.push_back(u);
74         }
75         for (int i = 0; i < n; i++) {
76             if (i + 1 < n) {
77                 record[i]->next = record[i + 1];
78             }
79             if (v[i] == 11111) {
80                 continue;
81             }
82             record[i]->random = record[v[i]];
83         }
84
85         solve(a);
86     }
87 }
88 void solve(Node* x) {
89     Node* res;
90     res = copyRandomList(x);
91     output(res);
92 }
93 void output(Node* x) {
94     Node* cur = x;
95     int cnt = 0;
96     map<Node*, int> mp;
97     while (cur != NULL) {
98         mp.insert(pair<Node*, int>(cur, cnt));
99         cnt++;

```

```

100     cur = cur->next;
101 }
102 cur = x;
103 while (cur != NULL) {
104     printf("%d ", cur->val);
105     if (cur->random == NULL)
106         printf("null\n");
107     else
108         printf("%d\n", mp[cur->random]);
109     cur = cur->next;
110 }
111 }
112 }
113
114 private:
115     int n;
116     Node* a;
117 };
118
119 int main() {
120     freopen("./assets/fipt.txt", "r", stdin);
121     freopen("./assets/fopt.txt", "w", stdout);
122
123     Solution sol;
124
125     sol.input();
126
127     return 0;
128 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.HashMap;
6  import java.util.LinkedList;
7  import java.util.Map;
8  import java.util.Scanner;
9
10 class Node {
11     int val;
12     Node next;
13     Node random;
14
15     public Node(int val) {
16         this.val = val;
17         this.next = null;
18         this.random = null;
19     }
20 }
21
22 class Solution {
23     public Node copyRandomList(Node head) {
24         if (head == null)
25             return null;
26
27         Node res = new Node(head.val);
28         Node cur_h = head;
29         Node cur_r = res;
30         Map<Node, Node> mp = new HashMap<Node, Node>();
31         mp.put(null, null);
32
33         while (cur_h != null) {
34             if (cur_h != head) {
35                 cur_r.next = new Node(cur_h.val);
36                 cur_r = cur_r.next;
37             }

```



```

38     mp.put(cur_h, cur_r);
39     cur_h = cur_h.next;
40 }
41
42 cur_h = head;
43 cur_r = res;
44 while (cur_h != null) {
45     cur_r.random = mp.get(cur_h.random);
46     cur_r = cur_r.next;
47
48     cur_h = cur_h.next;
49 }
50
51 return res;
52 }
53
54 public static void main(String[] args) throws FileNotFoundException {
55     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
56     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
57
58     System.setIn(fin);
59     System.setOut(fot);
60
61     Solution sol = new Solution();
62
63     sol.input();
64 }
65
66 public void input() {
67     Scanner in = new Scanner(System.in);
68     while (in.hasNext()) {
69         n = in.nextInt();
70
71         Node a = new Node(0);
72         Node u = a;
73         LinkedList<Integer> v = new LinkedList<Integer>();
74         LinkedList<Node> record = new LinkedList<Node>();
75
76         for (int i = 0; i < n; i++) {
77             if (i != 0) {
78                 u.next = new Node(0);
79                 u = u.next;
80             }
81             u.val = in.nextInt();
82
83             v.add(in.nextInt());
84             record.add(u);
85         }
86         for (int i = 0; i < n; i++) {
87             if (i + 1 < n) {
88                 record.get(i).next = record.get(i + 1);
89             }
90             if (v.get(i) == 11111) {
91                 continue;
92             }
93             record.get(i).random = record.get(v.get(i));
94         }
95
96         solve(a);
97     }
98
99     in.close();
100 }
101
102 public void solve(Node x) {
103     Node res;
104     res = copyRandomList(x);
105     output(res);
106 }

```

```

107     }
108
109     public void output(Node x) {
110         Node cur = x;
111         int cnt = 0;
112         Map<Node, Integer> mp = new HashMap<Node, Integer>();
113         while (cur != null) {
114             mp.put(cur, cnt);
115             cnt++;
116             cur = cur.next;
117         }
118         cur = x;
119         while (cur != null) {
120             System.out.print(cur.val + " ");
121             if (cur.random == null)
122                 System.out.println("null");
123             else
124                 System.out.println(mp.get(cur.random));
125
126             cur = cur.next;
127         }
128     }
129 }
130
131 private int n, m;
132 }

```

## B.16 Leetcode 141

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 typedef struct ListNode {
15     int val;
16     ListNode* next;
17     ListNode(int x) : val(x), next(NULL) {}
18 } ListNode;
19
20 class Solution {
21 public:
22     bool hasCycle(ListNode* head) {
23         if (NULL == head) return false;
24         ListNode *slow = head, *fast = head->next;
25         while (NULL != fast) {
26             if (slow == fast) return true;
27             slow = slow->next;
28             fast = fast->next;
29             if (NULL != fast) fast = fast->next;
30         }
31         return false;
32     }
33
34     void input(void) {
35         while (~scanf("%d %d", &n, &m)) {
36             int t;

```

```

37     scanf("%d", &t);
38     a = new ListNode(t);
39     ListNode* cur = a;
40
41     for (int i = 1; i < n; i++) {
42         scanf("%d", &t);
43         cur->next = new ListNode(t);
44         cur = cur->next;
45     }
46     ListNode* back = cur;
47     cur = a;
48     if (m >= 0) {
49         for (int i = 0; i < m - 1; i++) cur = cur->next;
50         back->next = cur;
51     }
52
53     solve(a);
54 }
55 }
56
57 void solve(ListNode* x) {
58     bool res;
59     res = hasCycle(x);
60     output(res);
61 }
62 void output(int x) { printf("%d\n", x); }
63
64 private:
65     int n, m;
66     ListNode* a;
67 };
68
69 int main() {
70     freopen("./assets/fipt.txt", "r", stdin);
71     freopen("./assets/fopt.txt", "w", stdout);
72
73     Solution sol;
74
75     sol.input();
76
77     return 0;
78 }
79

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      class ListNode {
9          int val;
10         ListNode next;
11
12         ListNode(int x) {
13             val = x;
14             next = null;
15         }
16     }
17
18     public boolean hasCycle(ListNode head) {
19         if (null == head)
20             return false;
21         ListNode slow = head, fast = head.next;
22         while (null != fast) {
23             if (slow == fast)

```

```

24     return true;
25     slow = slow.next;
26     fast = fast.next;
27     if (null != fast)
28         fast = fast.next;
29 }
30 return false;
31 }
32
33 public static void main(String[] args) throws FileNotFoundException {
34     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
35     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
36
37     System.setIn(fin);
38     System.setOut(fot);
39
40     Solution sol = new Solution();
41
42     sol.input();
43 }
44
45 public void input() {
46     Scanner in = new Scanner(System.in);
47     while (in.hasNext()) {
48         n = in.nextInt();
49         m = in.nextInt();
50         a = new ListNode(0);
51
52         int cnt = 0;
53         for (int i = 0; i < n; i++) {
54             if (cnt == 0) {
55                 a.val = in.nextInt();
56             } else {
57                 a.next = new ListNode(in.nextInt());
58             }
59         }
60
61         solve(a);
62     }
63
64     in.close();
65 }
66
67 public void solve(ListNode x) {
68     boolean res;
69     res = hasCycle(x);
70     output(res);
71 }
72
73
74 public void output(boolean x) {
75     System.out.println(x);
76 }
77
78 private int n, m;
79 private ListNode a;
80 }

```

## B.17 Leetcode 202

C++

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <cstdlib>
4 #include <cstring>
5 #include <iostream>

```

```

6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int cal(int x) {
17         int res = 0;
18         int t;
19         while (x != 0) {
20             t = x % 10;
21             x /= 10;
22             res += t * t;
23         }
24         return res;
25     }
26     bool isHappy(int n) {
27         if (n == 1) return true;
28         int slow = n, fast = cal(n);
29         while (slow != fast) {
30             if (slow == 1 || fast == 1) return true;
31             slow = cal(slow);
32             fast = cal(cal(fast));
33         }
34         return false;
35     }
36
37     void input(void) {
38         while (~scanf("%d", &n)) {
39             solve(n);
40         }
41     }
42
43     void solve(int x) {
44         bool res;
45         res = isHappy(x);
46         output(res);
47     }
48     void output(int x) { printf("%d\n", x); }
49
50 private:
51     int n, m;
52 };
53
54 int main() {
55     freopen("./assets/fipt.txt", "r", stdin);
56     freopen("./assets/fopt.txt", "w", stdout);
57
58     Solution sol;
59
60     sol.input();
61
62     return 0;
63 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public int cal(int x) {

```

```

9     int res = 0;
10    int t;
11    while (x != 0) {
12        t = x % 10;
13        x /= 10;
14        res += t * t;
15    }
16    return res;
17 }
18
19 public boolean isHappy(int n) {
20     if (n == 1)
21         return true;
22     int slow = n, fast = cal(n);
23     while (slow != fast) {
24         if (slow == 1 || fast == 1)
25             return true;
26         slow = cal(slow);
27         fast = cal(cal(fast));
28     }
29     return false;
30 }
31
32 public static void main(String[] args) throws FileNotFoundException {
33     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
34     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
35
36     System.setIn(fin);
37     System.setOut(fot);
38
39     Solution sol = new Solution();
40
41     sol.input();
42 }
43
44 public void input() {
45     Scanner in = new Scanner(System.in);
46     while (in.hasNext()) {
47         n = in.nextInt();
48         solve(n);
49     }
50
51     in.close();
52 }
53
54 public void solve(int x) {
55     boolean res;
56     res = isHappy(x);
57     output(res);
58 }
59
60
61 public void output(boolean x) {
62     System.out.println(x);
63 }
64
65 private int n, m;
66 private ListNode a;
67 }

```

## B.18 Leetcode 206

C++

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <cstdlib>

```

```

4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 typedef struct ListNode {
15     int val;
16     ListNode* next;
17     ListNode(int x) : val(x), next(NULL) {}
18 } ListNode;
19
20 class Solution {
21 public:
22     ListNode* reverseList(ListNode* head) {
23         if (head == NULL) return NULL;
24
25         ListNode* st = head;
26         ListNode* en = NULL;
27         ListNode* cur = head;
28         while (cur->next != NULL) {
29             cur = cur->next;
30         }
31         en = cur;
32         while (st != en) {
33             cur = st->next;
34             st->next = en->next;
35             en->next = st;
36             st = cur;
37         }
38         return en;
39     }
40     void input(void) {
41         while (~scanf("%d", &n)) {
42             a = new ListNode(0);
43             ListNode* u = a;
44             int x;
45
46             for (int i = 0; i < n; i++) {
47                 if (i != 0) {
48                     u->next = new ListNode(0);
49                     u = u->next;
50                 }
51
52                 scanf("%d", &u->val);
53             }
54
55             solve(a);
56         }
57     }
58     void solve(ListNode* x) {
59         ListNode* res;
60         res = reverseList(x);
61         output(res);
62     }
63     void output(ListNode* x) {
64         ListNode* cur = x;
65         while (cur != NULL) {
66             printf("%d\n", cur->val);
67             cur = cur->next;
68         }
69     }
70
71 private:
72     int n;

```

```

73  ListNode* a;
74  };
75
76  int main() {
77      freopen("./assets/fipt.txt", "r", stdin);
78      freopen("./assets/fopt.txt", "w", stdout);
79
80      Solution sol;
81
82      sol.input();
83
84      return 0;
85  }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public class ListNode {
9          int val;
10         ListNode next;
11
12         ListNode(int x) {
13             val = x;
14         }
15     }
16
17     public ListNode reverseList(ListNode head) {
18         if (head == null)
19             return null;
20
21         ListNode st = head;
22         ListNode en = null;
23         ListNode cur = head;
24         while (cur.next != null) {
25             cur = cur.next;
26         }
27         en = cur;
28         while (st != en) {
29             cur = st.next;
30             st.next = en.next;
31             en.next = st;
32             st = cur;
33         }
34         return en;
35     }
36
37     public static void main(String[] args) throws FileNotFoundException {
38         FileInputStream fin = new FileInputStream("./assets/fipt.txt");
39         PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
40
41         System.setIn(fin);
42         System.setOut(fot);
43
44         Solution sol = new Solution();
45
46         sol.input();
47     }
48
49     public void input() {
50         Scanner in = new Scanner(System.in);
51         while (in.hasNext()) {
52             n = in.nextInt();
53         }

```



```

54     ListNode a = new ListNode(0);
55     ListNode u = a;
56
57     for (int i = 0; i < n; i++) {
58         if (i != 0) {
59             u.next = new ListNode(0);
60             u = u.next;
61         }
62         u.val = in.nextInt();
63     }
64
65     solve(a);
66 }
67
68 in.close();
69 }
70
71 public void solve(ListNode x) {
72     ListNode res;
73     res = reverseList(x);
74     output(res);
75 }
76
77
78 public void output(ListNode x) {
79     ListNode cur = x;
80     while (cur != null) {
81         System.out.println(cur.val);
82         cur = cur.next;
83     }
84 }
85 }
86
87 private int n, m;
88 }

```

## B.19 Leetcode 209

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int minSubArrayLen(int s, vector<int>& nums) {
17         int res = nums.size();
18         int sz = res;
19         int sum = 0;
20         int u = 0, v = 0;
21         bool occur = false;
22
23         for (v = 0; v < sz; v++) {
24             sum += nums[v];
25             while (sum >= s && u <= v) {
26                 occur = true;
27                 res = min(res, v - u + 1);

```

```

28         sum -= nums[u];
29         u++;
30     }
31 }
32 if (occur == false) return 0;
33
34 return res;
35 }
36
37 void input(void) {
38     while (~scanf("%d %d", &n, &m)) {
39         int t;
40         for (int i = 0; i < n; i++) {
41             scanf("%d", &t);
42             a.push_back(t);
43         }
44         solve(m, a);
45         a.clear();
46     }
47 }
48 void solve(int s, vector<int>& x) {
49     int res;
50     res = minSubArrayLen(s, x);
51     output(res);
52 }
53 void output(int x) { printf("%d\n", x); }
54
55 private:
56     int n, m;
57     vector<int> a;
58 };
59
60 int main() {
61     freopen("./assets/fipt.txt", "r", stdin);
62     freopen("./assets/fopt.txt", "w", stdout);
63
64     Solution sol;
65
66     sol.input();
67
68     return 0;
69 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public int minSubArrayLen(int s, int[] nums) {
9          int res = nums.length;
10         int sz = res;
11         int sum = 0;
12         int u = 0, v = 0;
13         boolean occur = false;
14
15         for (v = 0; v < sz; v++) {
16             sum += nums[v];
17             while (sum >= s && u <= v) {
18                 occur = true;
19                 res = Math.min(res, v - u + 1);
20                 sum -= nums[u];
21                 u++;
22             }
23         }
24         if (occur == false)

```

```

25     return 0;
26
27     return res;
28 }
29
30 public static void main(String[] args) throws FileNotFoundException {
31     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
32     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
33
34     System.setIn(fin);
35     System.setOut(fot);
36
37     Solution sol = new Solution();
38
39     sol.input();
40 }
41
42 public void input() {
43     Scanner in = new Scanner(System.in);
44     while (in.hasNext()) {
45         n = in.nextInt();
46         m = in.nextInt();
47         a = new int[n];
48         for (int i = 0; i < n; i++)
49             a[i] = in.nextInt();
50         solve(m, a);
51     }
52
53     in.close();
54 }
55
56 public void solve(int s, int[] x) {
57     int res;
58     res = minSubArrayLen(s, x);
59     output(res);
60 }
61
62
63 public void output(int x) {
64     System.out.println(x);
65 }
66
67 private int n, m;
68 private int[] a;
69 }

```

## B.20 Leetcode 387

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Solution {
15 public:
16     int firstUniqChar(string s) {
17         int ans = -1;

```

```

18     int u, v;
19     for (char ch = 'a'; ch <= 'z'; ch++) {
20         u = s.find(ch);
21         v = s.rfind(ch);
22         if (u == v && u != -1) {
23             if (ans == -1) {
24                 ans = u;
25             } else {
26                 if (u < ans) ans = u;
27             }
28         }
29     }
30     return ans;
31 }
32 void input(void) {
33     while (cin >> str) {
34         solve(str);
35     }
36 }
37 void solve(string s) {
38     int res;
39     res = firstUniqChar(s);
40     output(res);
41 }
42 void output(int id) { printf("%d\n", id); }
43
44 private:
45     int n, m, t;
46     string str;
47 };
48
49 int main() {
50     freopen("./assets/fipt.txt", "r", stdin);
51     freopen("./assets/fopt.txt", "w", stdout);
52
53     Solution sol;
54
55     sol.input();
56
57     return 0;
58 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public int firstUniqChar(String s) {
9          int ans = -1;
10         int u, v;
11         for (char ch = 'a'; ch <= 'z'; ch++) {
12             u = s.indexOf(ch);
13             v = s.lastIndexOf(ch);
14             if (u == v && u != -1) {
15                 if (ans == -1) {
16                     ans = u;
17                 } else {
18                     if (u < ans)
19                         ans = u;
20                 }
21             }
22         }
23         return ans;
24     }
25 }

```

```

26 public static void main(String[] args) throws FileNotFoundException {
27     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
28     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
29
30     System.setIn(fin);
31     System.setOut(fot);
32
33     Solution sol = new Solution();
34
35     sol.input();
36 }
37
38 public void input() {
39     Scanner in = new Scanner(System.in);
40     while (in.hasNext()) {
41         String str = in.nextLine();
42         solve(str);
43     }
44
45     in.close();
46 }
47
48 public void solve(String s) {
49     int res;
50     res = firstUniqChar(s);
51     output(res);
52 }
53
54 public void output(int idx) {
55     System.out.println(idx);
56 }
57
58 private int n, m, t;
59 private int u, v;
60 private int[] numbers = new int[10000];
61 }

```

## B.21 Leetcode 876

C++

```

1  #include <algorithm>
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
6  #include <map>
7  #include <queue>
8  #include <stack>
9  #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 typedef struct ListNode {
15     int val;
16     ListNode* next;
17     ListNode(int x) : val(x), next(NULL) {}
18 } ListNode;
19
20 class Solution {
21 public:
22     public:
23     ListNode* middleNode(ListNode* head) {
24         if (NULL == head) return NULL;
25         ListNode* slow = head;
26         ListNode* fast = head->next;

```

```

27     while (fast != NULL) {
28         slow = slow->next;
29         fast = fast->next;
30         if (fast == NULL) {
31             return slow;
32         } else {
33             fast = fast->next;
34         }
35     }
36     return slow;
37 }
38
39 void input(void) {
40     while (~scanf("%d", &n)) {
41         a = new ListNode(0);
42         ListNode* cur = a;
43         for (int i = 0; i < n; i++) {
44             int t;
45             scanf("%d", &t);
46             if (i == 0) {
47                 cur->val = t;
48             } else {
49                 cur->next = new ListNode(t);
50                 cur = cur->next;
51             }
52         }
53         solve(a);
54     }
55 }
56
57 void solve(ListNode* x) {
58     ListNode* res;
59     res = middleNode(x);
60     output(res);
61 }
62 void output(ListNode* x) {
63     ListNode* cur = x;
64     while (cur != NULL) {
65         printf("%d\n", cur->val);
66         cur = cur->next;
67     }
68 }
69
70 private:
71     int n, m;
72     ListNode* a;
73 };
74
75 int main() {
76     freopen("./assets/fipt.txt", "r", stdin);
77     freopen("./assets/fopt.txt", "w", stdout);
78
79     Solution sol;
80
81     sol.input();
82
83     return 0;
84 }

```

## Java

```

1  import java.io.FileInputStream;
2  import java.io.FileNotFoundException;
3  import java.io.FileOutputStream;
4  import java.io.PrintStream;
5  import java.util.Scanner;
6
7  class Solution {
8      public class ListNode {

```

```

9      int val;
10     ListNode next;
11
12     ListNode(int x) {
13         val = x;
14     }
15 }
16
17 public ListNode middleNode(ListNode head) {
18     if (null == head)
19         return null;
20     ListNode slow = head;
21     ListNode fast = head.next;
22     while (fast != null) {
23         slow = slow.next;
24         fast = fast.next;
25         if (fast == null) {
26             return slow;
27         } else {
28             fast = fast.next;
29         }
30     }
31     return slow;
32 }
33
34 public static void main(String[] args) throws FileNotFoundException {
35     FileInputStream fin = new FileInputStream("./assets/fipt.txt");
36     PrintStream fot = new PrintStream(new FileOutputStream("./assets/fopt.txt"));
37
38     System.setIn(fin);
39     System.setOut(fot);
40
41     Solution sol = new Solution();
42
43     sol.input();
44 }
45
46 public void input() {
47     Scanner in = new Scanner(System.in);
48     while (in.hasNext()) {
49         n = in.nextInt();
50         a = new ListNode(0);
51         ListNode cur = a;
52         for (int i = 0; i < n; i++) {
53             if (i == 0) {
54                 cur.val = in.nextInt();
55             } else {
56                 cur.next = new ListNode(in.nextInt());
57                 cur = cur.next;
58             }
59         }
60         solve(a);
61     }
62
63     in.close();
64 }
65
66 public void solve(ListNode x) {
67     ListNode res;
68     res = middleNode(x);
69     output(res);
70 }
71
72
73 public void output(ListNode x) {
74     ListNode cur = x;
75     while (cur != null) {
76         System.out.println(cur.val);
77         cur = cur.next;

```

```
78     }  
79 }  
80  
81 private int n, m;  
82 private ListNode a;  
83 }
```