Stack

Q17

利用 系统提供的 list 相关类型，

比如 java 里的 List / ArrayList，js 里的 Array

实现 Stack

Stack

结构

source: List<Int>

借用系统的 list 结构 及 api。

API

push(value: Int)

进栈

pop(): Int

出栈

top(): Int

返回栈顶元素，但是不出栈

size(): Int

返回栈的大小

大鱼吃小鱼

Q18

微软面试题

背景

鱼与尺寸

在一个鱼塘里 有一堆鱼 $f_1 ... f_n$

第 i 只鱼的大小为 s_i

每只鱼刚开始的大小均为 1

吃鱼规则

当一只鱼 f_a 的尺寸 s_a 大于等于 另外一只鱼 f_b 的尺寸 s_b 时，

f_a 可以吃掉 f_b。

吃掉后 f_a 的尺寸变为 $s_a + s_b$，而 f_b 的尺寸变为 0

要求

输入

给定 n 只鱼，和 一个指令数组

指令数组 的每个元素为一个2个元素的数组

比如

```
1, 1
1, 2
3, 1
```

每一行的为指令，代表 左边数字的那只鱼 要吃掉 右边的那只鱼

输出

返回一个 n 尺寸的数组，

如果该鱼没有被吃，则为 0。

如果被吃了，则显示为 现在它在哪只鱼的肚子里

案例

假设 n 是 3

假设指令数组为

```
1, 1
1, 2
3, 2
3, 1
```

第一条指令不能执行，因为自己不能吃自己

第二条指令可以执行，执行后，第一条鱼的大小为 2， 第二条鱼的大小为 0

第三条指令不能执行，因为 第2条鱼 已经被吃到 第1条鱼的肚子里了

第四条指令不能执行，因为 第3条鱼的尺寸 小于 第1条鱼的尺寸

目前为止鱼 1 和 3 还活着，尺寸分别是 2，1

鱼 2 在 1 的肚子里

需要返回

```
[0, 1, 0]
```

API

start(n: Int, commands: Int) : Int

ArrayList

Q15

根据指定的结构 和 方法要求 实现 ArrayList

假设 ArrayList 里存储的元素 是 Int

结构

source: Int

指向基础数组的指针

我们可以设定 ArrayList 刚开始的时候，`source` 是一个指向 尺寸 为 4 的基础数组。

随着 对元素的增加，你自己需要 进行扩容操作。

size: Int

多少个数组内的空间，真正被占用

API

add(value: Int)

往集合的最后一个位置 增加一个元素 `value`

insert(index: Int, value: Int)

往集合的 `index` 位置插入元素 `value`

样例：

```
list is <6, 7, 3, 8, 4>
list.insert(2, 9);
list is <6, 7, 9, 3, 8, 4>
list.insert(6, 1);
list is <6, 7, 9, 3, 8, 4, 6>
```

合法性检查：

需要检测 `index` 的合法性。

`index` 应该 大于等于 0，且小于等于 当前元素个数

如果 `index` 非法，则不做任何操作

get(index: Int): Int

返回集合中 `index` 的位置

size(): Int

返回 ArrayList 里有多少个元素

toString(): String

返回字符串

格式：

如果数组内有一些元素，返回

```
"<6, 7, 3, 8, 4>"
```

如果数组内没有元素，返回

```
"<>"
```

双向链表

Q16

根据指定的结构 和 方法要求 实现 LinkedList

假设 LinkedList 里存储的元素 是 Int

LinkedListNode

结构

prev: LinkedListNode

指向上一个节点的指针

next: LinkedListNode

指向下一个节点的指针

value: Int

当前节点所存储的值

LinkedList

结构

head: LinkedListNode

指向头节点的指针

tail: LinkedListNode

指向尾节点的指针

size: Int

对链表尺寸进行缓存

API

add(value: Int)

往集合的最后一个位置 增加一个元素 `value`

insert(index: Int, value: Int)

往集合的 `index` 位置插入元素 `value`

样例：

```
list is <6, 7, 3, 8, 4>
list.insert(2, 9);
list is <6, 7, 9, 3, 8, 4>
list.insert(6, 1);
list is <6, 7, 9, 3, 8, 4, 6>
```

合法性检查：

需要检测 `index` 的合法性。

`index` 应该 大于等于 0，且小于等于 当前元素个数

如果 `index` 非法，则不做任何操作

get(index: Int): Int

返回集合中 `index` 的位置

size(): Int

返回 ArrayList 里有多少个元素

toString(): String

返回字符串

格式：

如果数组内有一些元素，返回

```
"<6, 7, 3, 8, 4>"
```

如果数组内没有元素，返回

```
"<>"
```