编程题第一题 版本号 AC 80%

```
package middleLinkCode;

import java.util.Scanner;

public class xinlang1 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		// 3, 4.3.5.4 2.10.3 2.4
		Scanner sc = new Scanner(System.in);
		String[] s1 = sc.nextLine().split(" ");

		System.out.println(getMinVersion(s1));
		sc.close();
	}

	public static String getMinVersion(String[] list) {
		//  在这里编写代码
		int len = list.length;
		if (len == 0 || list == null)
			return null;
		String min = "";
		for (int i = 0; i < len - 1; i++) {
			if (cal(list[i], list[i + 1]) == 1) {
				min = list[i + 1];
			} else if (cal(list[i], list[i + 1]) == -1) {
				min = list[i];
			} else
				min = list[i];
		}
		return min;
	}

	public static int cal(String s1, String s2) {
		String[] t1 = s1.split("\\.");
		String[] t2 = s2.split("\\.");
		int len1 = t1.length;
		int len2 = t2.length;
		int len = Math.min(len1, len2);
		int i;
		for (i = 0; i < len; i++) {
			int a = Integer.parseInt(t1[i]);
			int b = Integer.parseInt(t2[i]);
```

```
                if (a > b) {
                    return 1;
                } else if (a < b) {
                    return -1;
                }
            }
            if (len1 > len2) {
                for (int j = i; j < len1; j++) {
                    int temp = Integer.parseInt(t1[j]);
                    if (temp != 0)
                        return 1;
                }
                return 0;
            } else if (len1 < len2) {
                for (int j = i; j < len2; j++) {
                    int temp = Integer.parseInt(t2[j]);
                    if (temp != 0)
                        return 1;
                }
            }
            return 0;
        }
}
```

第二题 AC 100% 实现 LRU
可以借助 hashmap

```
package middleLinkCode;

import java.util.HashMap;

public class Solution {

    private Node curH = new Node();
    private Node curT = new Node();
    private int capacity;
    private int size;
    private HashMap<Integer, Node> tempMap = new HashMap<>();

    private void add(Node node) {
        Node temp = curH.next;
        curH.next = node;
        node.pre = curH;
        node.next = temp;
        temp.pre = node;
```

```
}

private void del(Node node) {
        Node p = node.pre;
        Node q = node.next;
        p.next = q;
        q.pre = p;
        node.pre = null;
        node.next = null;
}

public Solution(int capacity) {
        curH.next = curT;
        curT.pre = curH;
        this.capacity = capacity;
        size = 0;
}

public int get(int key) {
        Node p = tempMap.get(key);
        if (p == null)
                return -1;
        del(p);
        add(p);
        return p.value;
}

public void put(int key, int value) {
        Node p = tempMap.get(key);
        if (p != null) {
                p.value = value;
                del(p);
                add(p);
        } else {
                if (size < capacity)
                        size++;
                else {
                        Node q = curT.pre;
                        tempMap.remove(q.key);
                        del(q);
                }
                Node t = new Node(key, value);
                add(t);
                tempMap.put(key, t);
```

```
            }
        }

        private class Node {
            private int key;
            private int value;
            private Node pre;
            private Node next;

            public Node() {
            }

            public Node(int key, int value) {
                this.key = key;
                this.value = value;
            }
        }
}
```