题目描述：

手上有一副扑克牌，每张牌按牌面数字记分（J=11,Q=12,K=13，没有大小王），出牌时按照以下规则记分：

■ 出单张，记牌面分数，例如出一张 2，得分为 2

■ 出对或 3 张，记牌面分数总和再 x2，例如出 3 张 3，得分为(3+3+3)x2=18

■ 出 5 张顺，记牌面分数总和再 x2，例如出 34567 顺，得分为(3+4+5+6+7)x2=50

■ 出 4 张炸弹，记牌面分数总和再 x3，例如出 4 张 4，得分为 4x4x3=48

求出一副牌最高的得分数

输入描述：

按顺序排好的一副牌，最少 1 张，最多 15 张。

1-9 输入为数字 1-9，10 输入为数字 0，JQK 输入为大写字母 JQK。

无需考虑输入非法的情况，例如输入字符不在[0-9JQK]范围或某一张牌超过 4 张。

输出描述：

最高的得分数

补充说明：

积分规则中没有的出牌方式不支持，例如不支持 3 带 1、4 带 2，不支持 5 张以上的顺，且 10JQKA（0JQK1）不算顺。

示例 1

输入：

```
33445677
```

输出：

```
67
```

说明：

出对 3、对 4、对 7，单张 5、6，得分为 67；出 34567 顺，再出单张 3、4、7，得分为 64

因此最高得分是按对出，可得到最高分 67，输出结果 67

```
import java.util.*;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        HashMap<Character, Integer> map = new
        HashMap<Character, Integer>();
        map.put('1', 1);
        map.put('2', 2);
        map.put('3', 3);
        map.put('4', 4);
        map.put('5', 5);
        map.put('6', 6);
        map.put('7', 7);
        map.put('8', 8);
        map.put('9', 9);
        map.put('0', 10);
        map.put('J', 11);
        map.put('Q', 12);
        map.put('K', 13);
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        String str = in.nextLine();
        int[] arr1 = new int[14];
        int[] arr2 = new int[14];
        for (int i = 0; i < str.length(); i++) {
            arr1[map.get(str.charAt(i))]++;
            arr2[map.get(str.charAt(i))]++;
        }
        int max1 = 0, max2 = 0,max3=0;
        for (int i = 1; i < 14; i++) {
            if (arr1[i] > 1) {
                if (arr1[i] == 4) {
                    max1 += arr1[i] * i * 3;
                } else {
                    max1 += arr1[i] * i * 2;
                }
            } else {
```

```
                max1 += arr1[i] * i;
        }
}
int strat = map.get(str.charAt(0)), end = map.get(str.charAt(str.length() - 1));
if (end - strat < 4) {
        System.out.println(max1);
        return;
}
int l1 = strat, r1 = l1 + 4;
while (r1 <= end) {
        int flag = 1;
        for (int i = l1; i <= r1; i++) {
                if (arr1[i] == 0||arr1[i]==4) {
                        flag = 0;
                }
        }
        if (flag == 1) {
                max2 += (l1 + r1) * 5;
                for (int j = l1; j <= r1; j++) {
                        arr1[j]--;
                }
        }
        l1++;
        r1++;
}
for (int i = 1; i < 14; i++) {
        if (arr1[i] > 1) {
                if (arr1[i] == 4) {
                        max2 += arr1[i] * i * 3;
                } else {
                        max2 += arr1[i] * i * 2;
                }
        } else {
                max2 += arr1[i] * i;
        }
}
int r2=end,l2=r2-4;
while (l2 >= strat) {
        int flag = 1;
        for (int i = l2; i <= r2; i++) {
                if (arr2[i] == 0) {
                        flag = 0;
                }
        }
```

```
                if (flag == 1) {
                    max3 += (l2 + r2) * 5;
                    for (int j = l2; j <= r2; j++) {
                        arr2[j]--;
                    }
                }
                l2--;
                r2--;
            }
            for (int i = 1; i < 14; i++) {
                if (arr2[i] > 1) {
                    if (arr2[i] == 4) {
                        max3 += arr2[i] * i * 3;
                    } else {
                        max3 += arr2[i] * i * 2;
                    }
                } else {
                    max3 += arr2[i] * i;
                }
            }
            int ans=Math.max(max1, max2);
            ans=Math.max(ans, max3);
            System.out.println(ans);
        };
}
```