# Assign #3: Oct Mock Exam暨选做题目满百

Updated 1537 GMT+8 Oct 10, 2024

2024 fall, Complied by Hongfei Yan==（请改为同学的姓名、院系）==

**姓名：李彦臻**
**学号：2300010821**
**学院：数学科学学院**
**月考成绩：AC5**

**闫老师好，实在不好意思，我这两天感冒了，发烧了身体很不舒服，没去上课也忘记了 ddl 在哪一天，迟交了一天，现在把作业补给您，谢谢！**

**说明：**

1）Oct月考： AC6==（请改为同学的通过数）== 。考试题目都在“题库（包括计概、数算题目）”里面，按照数字题号能找到，可以重新提交。作业中提交自己最满意版本的代码和截图。

2）请把每个题目解题思路（可选），源码 Python, 或者 C++/C（已经在 Codeforces/Openjudge 上 AC），截图（包含 Accepted, 学号），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用 word）。AC 或者没有 AC，都请标上每个题目大致花费时间。

3）提交时候先提交 pdf 文件，再把 md 或者 doc 文件上传到右侧“作业评论”。Canvas 需要有同学清晰头像、提交文件有 pdf、作业评论有 md 或者 doc。

4）如果不能在截止前提交作业，请写明原因。

## 1. 题目

### E28674:《黑神话：悟空》之加密

http://cs101.openjudge.cn/practice/28674/

思路：这道题目只要知道了 ord 和 chr 这两个函数就很好做。这两个函数的简要介绍：ord 把每个“国际标准字符”转换为数，chr 则反过来~
其中，需要记住的是 **A~Z 分别对应 65~90，a~z 分别对应 97~122**；但需要注意：ord 只能对单个字符使用；chr 的定义域则是 0~110000，不要超出定义域了。
一旦把字符转换为了数字之后，只需要先根据大小写分类讨论，然后再考虑其平移之后关于 26 的模，并放置到大写 or 小写的对应区间即可！

代码

```python


```

```
k=int(input())
t=k % 26
alphabets=list(map(str,input()))
n=len(alphabets)
result=[]

for i in range(n):
    j=ord(alphabets[i])
    if j <= 90:#为大写字母
        if j - t >= 65:
            result.append(chr(j - t))
        else:
            result.append(chr(j - t + 26))
    else:#为小写字母
        if j - t >= 97:
            result.append(chr(j - t))
        else:
            result.append(chr(j - t + 26))

print("".join(map(str,result)))
```

代码运行截图 ==（至少包含有"Accepted"）==

**#46523706提交状态**

查看 提交 统计 提问

状态: Accepted

基本信息

#: 46523706
题目: 28674
提交人: 24n2300010821
内存: 3648kB
时间: 20ms
语言: Python3
提交时间: 2024-10-16 17:00:31

源代码

```python
#介绍：ord把每个"国际标准字符"转换为数，chr则反过来~
#其中，A~Z分别对应65~90，a~z分别对应97~122；
#注意：ord只能对单个字符使用；chr定义域为0~110000

k=int(input())
t=k % 26
alphabets=list(map(str,input()))
n=len(alphabets)
result=[]

for i in range(n):
    j=ord(alphabets[i])
    if j <= 90:#为大写字母
        if j - t >= 65:
            result.append(chr(j - t))
        else:
            result.append(chr(j - t + 26))
    else:#为小写字母
        if j - t >= 97:
            result.append(chr(j - t))
        else:
            result.append(chr(j - t + 26))

print("".join(map(str,result)))
```

### E28691: 字符串中的整数求和

http://cs101.openjudge.cn/practice/28691/

思路：这道题目颇为简单，只需要把这两个字符串转化为列表形式，再把里面的每个元素都单独提取出来，然后再取我们想要的前两位并利用 int（）函数转化为整数来相加即可~

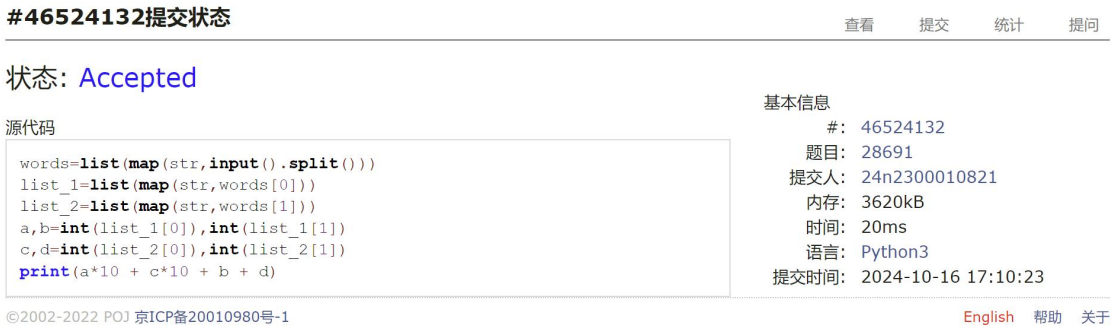
代码

```python

```

```
words=list(map(str,input().split()))
list_1=list(map(str,words[0]))
list_2=list(map(str,words[1]))
a,b=int(list_1[0]),int(list_1[1])
c,d=int(list_2[0]),int(list_2[1])
print(a*10 + c*10 + b + d)
```

代码运行截图 ==（至少包含有"Accepted"）==




### M28664: 验证身份证号

http://cs101.openjudge.cn/practice/28664/

思路：这道题的思路也较为简单：先把身份证号内的数字以 str 的形式放入一个列表，并把 X 统一为 10，这样方便后续的处理；接着把所有的权重放入一个列表，方便计算权重与身份证的对应位数相乘；相乘的时候**一定别忘了把 numbers 内的元素用 int（）转换为整数**，不然无法相乘！接下来就是计算总和，并求出它模 11 的余数；最后再与结果进行比对即可。（为了方便，可以**把正确结果放进一个列表**，算出余数之后把身份证号的最后一位**直接与正确结果列表的第余数位进行对比**即可！但别忘了把正确结果列表里的 X 也转换成 10~）

代码

```python

```

```
def verify(numbers):
    weight=[7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2]
    sum=0
    for i in range(17):
        sum += int(numbers[i])*weight[i]#记得对 numbers 内元素转换为数字
    k=sum % 11
    verify_list=[1,0,10,9,8,7,6,5,4,3,2]#把‘X’视为 10，方便比较大小
```

```
    if numbers[17] =='X':
        numbers[17]=10
    if verify_list[k] == int(numbers[17]):#记得转换格式！不然无法比较 str 的大小
        result.append("YES")
    else:
        result.append("NO")

n=int(input())
result=[]
for j in range(n):
    numbers=list(map(str,input()))
    verify(numbers)

for word in result:
    print(word)
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

#46525097提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def verify(numbers):
    weight=[7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2]
    sum=0
    for i in range(17):
        sum += int(numbers[i])*weight[i]#记得对numbers内元素转换为数字
    k=sum % 11
    verify_list=[1,0,10,9,8,7,6,5,4,3,2]#把'X'视为10，方便比较大小
    if numbers[17] =='X':
        numbers[17]=10
    if verify_list[k] == int(numbers[17]):#记得转换格式！不然无法比较str的大
        result.append("YES")
    else:
        result.append("NO")

n=int(input())
result=[]
for j in range(n):
    numbers=list(map(str,input()))
    verify(numbers)

for word in result:
    print(word)
```

基本信息

#: 46525097
题目: 28664
提交人: 24n2300010821
内存: 3652kB
时间: 20ms
语言: Python3
提交时间: 2024-10-16 17:30:21

### M28678: 角谷猜想

http://cs101.openjudge.cn/practice/28678/

思路：这道题目与每日选做里的“how old are you”那道题几乎一模一样，十分简单；大致方法就是使用 while 循环，只要 n 还不等于 1，就根据 n 的奇偶性分别进行对应的操作；并在**每一步操作的时候把那个操作的对应等式放入一个专门记录每一步操作的列表**即可。最后在 n 等于 1 并且循环结束的时候，往列表里扔一个'End'即可
接下来，把这个列表打印出来就立刻做完了~

代码

```python
```

```
n=int(input())
steps=[]
while n != 1:
    if n % 2 == 1:
        steps.append(f"{n}*3+1={n * 3 + 1}")
        n=n * 3 + 1
    else:
        steps.append(f"{n}/2={n//2}")
        n=n // 2#要用//防止出现 float
steps.append('End')
for step in steps:
    print(step)
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

### M28700: 罗马数字与整数的转换

http://cs101.openjudge.cn/practice/28700/

思路：这道题本质难度不高，主要是讨论起来比较繁琐。
第一步：**创建两个函数**，分别把罗马数字变成数字，和把数字变成罗马数字的函数；并且在开始输出之前先对输入的是罗马数字还是正常数字进行判断，从而决定执行哪个函数；
第二步：写把罗马数字变成数字的函数，方法主要是**把罗马数字拆成若干个小块**，算出它们对应的数并相加；其中，从左往右先出现的 M.D.L.V 一定为单独一个模块；而若先出现 I 和 X，要根据后面的字母分类讨论，可能 I.X.C 是单独的，也可能是 IV.IX;XL.XC;CD.CM!
第三步：写把数字变成罗马数字的函数，方法是先把千十百个位算出来，方便后续操作；随后**根据每一位的数是 0~3 还是 4 还是 5 还是 6~8 还是 9 单独进行讨论~**

##### 代码

```python
#

```

```
def rom_to_num(a):#方法：把罗马数字拆成若干个小块，
    #其中：先出现的 M.D.L.V 一定为单独一个模块；
    #先出现 I 和 X，要根据后面的字母分类讨论，
    #可能 I.X.C 是单独的，也可能是 IV.IX;XL.XC;CD.CM!
    alphabets=list(map(str,a))
    n=len(alphabets)
    num=0
    i=0
    while i < n:
        if alphabets[i] == 'M':#先把这四个简单情况搞定
            num += 1000
            i += 1
        elif alphabets[i] == 'D':
            num += 500
            i += 1
        elif alphabets[i] == 'L':
            num += 50
            i += 1
        elif alphabets[i] == 'V':
            num += 5
            i += 1
        elif alphabets[i] == 'C':#接下来处理三种复杂情况
```

```
            if i == n-1:#先把可能导致问题的情况扔掉
                num += 100
                i += 1
            elif alphabets[i+1] == 'D':
                num += 400
                i += 2#可以直接把这一个数组'CD'扔掉了~
            elif alphabets[i+1] == 'M':
                num += 900
                i += 2
            else:#不是 L 也不是 C，则为单独的 X!
                num += 100
                i += 1
        elif alphabets[i] == 'X':
            if i == n-1:
                num += 10
                i += 1
            elif alphabets[i+1] == 'L':
                num += 40
                i += 2#可以直接把这一个数组'XL'扔掉了~
            elif alphabets[i+1] == 'C':
                num += 90
                i += 2
            else:#不是 L 也不是 C，则为单独的 X!
                num += 10
                i += 1
        else:#最后一种情况：为 I~
            if i == n-1:
                num += 1
                i += 1
            elif alphabets[i+1] == 'V':
                num += 4
                i += 2#可以直接把这一个数组'IV'扔掉了~
            elif alphabets[i+1] == 'X':
                num += 9
                i += 2
            else:#不是 V 也不是 X，则为单独的 I!
                num += 1
                i += 1
    print(num)

def num_to_rom(a):#方法：根据每一位的数单独讨论~
    x=a // 1000#先把千十百个位算出来，方便后续操作
    y=(a - x*1000) // 100
    z=(a - x*1000 - y*100) // 10
    t=(a - x*1000 - y*100 - z*10) // 1
    letters=[]
    for i in range(x):
```

```
    letters.append('M')#先往里面添加 x 个 M，把千位数搞定

if y == 5:#挨个儿处理百位数的情况
    letters.append('D')
elif 6 <= y <= 8:
    letters.append('D')
    for i in range(y-5):#添加 y-5 个 C
        letters.append('C')
elif y == 9:
    letters.append('C')
    letters.append('M')
elif y == 4:
    letters.append('C')
    letters.append('D')
else:#y 在 0~3 之间，则添加 y 个 C
    for i in range(y):
        letters.append('C')

if z == 5:#挨个儿处理十位数的情况
    letters.append('L')
elif 6 <= z <= 8:
    letters.append('L')
    for i in range(z-5):#添加 z-5 个 X
        letters.append('X')
elif z == 9:
    letters.append('X')
    letters.append('C')
elif z == 4:
    letters.append('X')
    letters.append('L')
else:#z 在 0~3 之间，则添加 z 个 X
    for i in range(z):
        letters.append('X')

if t == 5:#挨个儿处理十位数的情况
    letters.append('V')
elif 6 <= t <= 8:
    letters.append('V')
    for i in range(t-5):#添加 t-5 个 I
        letters.append('I')
elif t == 9:
    letters.append('I')
    letters.append('X')
elif t == 4:
    letters.append('I')
    letters.append('V')
else:#t 在 0~3 之间，则添加 t 个 I
```

```
        for i in range(t):
            letters.append('I')
    #大功告成!
    print("".join(map(str,letters)))

a=input()
try:
    a=int(a)#如果是整数，才会执行 try 剩下的部分（第一个函数）
    num_to_rom(a)
except ValueError:
    #无法转换为整数则说明输入的是罗马数字，于是执行第二个函数
    rom_to_num(a)
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

**#46535941提交状态**

查看 提交 统计 提问

状态: Accepted

源代码

```
def rom_to_num(a):#方法：把罗马数字拆成若干个小块，
    #其中：先出现的M.D.L.V一定为单独一个模块；
    #先出现I和X，要根据后面的字母分类讨论，
    #可能I.X.C是单独的，也可能是IV.IX;XL.XC;CD.CM!
    alphabets=list(map(str,a))
    n=len(alphabets)
    num=0
    i=0
    while i < n:
        if alphabets[i] == 'M':#先把这四个简单情况搞定
            num += 1000
            i += 1
        elif alphabets[i] == 'D':
            num += 500
            i += 1
        elif alphabets[i] == 'L':
            num += 50
            i += 1
        elif alphabets[i] == 'V':
            num += 5
            i += 1
        elif alphabets[i] == 'C':#接下来处理三种复杂情况
            if i == n-1:#先把可能导致问题的情况扔掉
                num += 100
                i += 1
            elif alphabets[i+1] == 'D':
                num += 400
                i += 2#可以直接把这一个数组'CD'扔掉了~
```

基本信息

| | |
|---|---|
| #: | 46535941 |
| 题目: | 28700 |
| 提交人: | 24n2300010821 |
| 内存: | 3876kB |
| 时间: | 23ms |
| 语言: | Python3 |
| 提交时间: | 2024-10-16 22:24:07 |

### *T25353: 排队 （选做）

http://cs101.openjudge.cn/practice/25353/

思路：不太会。。（未做 qwq）

代码

```python

```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

## 2. 学习总结和收获

==如果作业题目简单，有否额外练习题目，比如：OJ“计概 2024fall 每日选做”、CF、LeetCode、洛谷等网站题目。==

自从对 Python 的基本语法结构初步熟悉了，逐渐起飞了之后，我也从一开始的慢慢追赶每日选做题的进度，到后来已经能稳步做出来绝大部分每日选做题；而那些比较难的题目也在花半了个小时或者 1 个多小时在一次次的尝试和与 AI 的交流中 ac 了之后，十分有成就感，也让我体会到了编程以及计算概论这门课的乐趣～
除了自己能力的逐渐长进以外，我也在不断的做题过程中学到了许多新的方法和思路，比如如果 TLE，可以把 list 改成 set，这样判断一个数在不在一个东西内可以大幅减少时间复杂度！

判断一个数在不在一个长度为n的列表里的时间复杂度是多少

判断一个数是否在一个长度为 $n$ 的列表中，时间复杂度取决于列表的类型和所使用的方法：

1. **列表（List）**：
   - 在 Python 中，使用 `in` 关键字或者 `list.index()` 方法检查一个数是否在列表中，最坏情况下需要遍历整个列表。
   - **时间复杂度**：$O(n)$
2. **集合（Set）**：
   - 如果列表被转换为集合（`set`），可以利用集合的哈希表特性来快速查找。
   - **时间复杂度**：$O(1)$（平均情况下）
3. **字典（Dictionary）**：
   - 如果列表中的元素作为字典的键存储，查找时间复杂度也为 $O(1)$（平均情况下）。

还有 set 的一些注意事项，比如不能用{}创建空集合，因为系统会自动把{}识别为字典！以及

set 里面没有 append，只能 add；但好处是如果 add 了重复元素，则并无影响~
还学到了如何对字典中的值进行排序：

```
sorted_dict=sorted(weight_price.items(),key=lambda item:item[1],reverse=True)
#把weight_price按照单位价值排序~（上一行含义：
#第一步：weight_price.items()返回字典中所有键值对，结果是一个由键值对元组构成的列表！
#第二步：key是sorted()的一个参数，指定根据什么标准进行排序
#第三步：lambda item:item[1]是一个函数，item代表传入的每一个键值对元组，
#而这里的item[1]，表示取元组的第二个元素（即字典的value）！
#总之，这个key函数确保sorted()按照每个元组的第二个元素（也就是value）来排序~
#第四步：reverse是一个参数，设置为True则sorted()会按降序排列；设为False或默认值，则为升序
#到了最后，返回的是一个按照value降序排序的列表，而不是字典本身~
```

（这取自我某道 ac 了的题目里的代码的注释）
实际上，这个方法也适用于对“列表里的列表”里的某个固定位置的元素进行排序！（对矩阵很有效）并且，如果要对列表里的每个列表的第 k 个位置的元素排序，写 item[k-1]即可！
不仅如此，我还在一个 wa 中学到了不要把一个 list 写为 [[0] * c2] * c1 ；这是因为当你写 [[0] * c2] * c1 时，Python 实际上只创建了一个长度为 c2 的列表 [0] * c2，然后将它的引用复制了 c1 次。这样导致所有行实际上引用的是同一个列表，即每一行都指向同一个对象。因此，当你修改矩阵的某一行的元素时，其他所有行也会同时改变，因为它们都指向同一个列表；这样会导致很严重的错误！

final_matrix=[[0]*c2]*c1

在 Python 中，使用 `[[0] * c2] * c1` 来创建二维列表（矩阵）的方式可能会产生**意想不到的问题**。这是因为它创建的是**同一个列表的多次引用**，而不是每一行都创建一个独立的列表。

### 具体解释：

当你写 `[[0] * c2] * c1` 时，Python 实际上只创建了一个长度为 `c2` 的列表 `[0] * c2`，然后将它的引用复制了 `c1` 次。这样导致所有行实际上引用的是**同一个列表**，即**每一行都指向同一个对象**。

因此，当你修改矩阵的某一行的元素时，**其他所有行也会同时改变**，因为它们都指向同一个列表。

正确的写法例如：

### 正确的创建方法：

为了避免这个问题，你应该使用列表推导式来确保每一行都是一个独立的列表。

**正确的方式：**

python 复制代码

```
final_matrix = [[0] * c2 for _ in range(c1)]
```

当然，正确的写法有很多种，这只是最简单的一种~
总之，这一段时间我在尝试 ac 一道道题目并与 AI 交流的过程中，以及在参考课程群里许多同学的答疑和 ac 代码中，学到了很多新的知识，有很多收获！希望能继续加油，早日 ac6～