HW#7

经济学院 刘安澜 1700015495

266A. 122A. Lucky Division

brute force/number theory, 1000,
https://codeforces.com/problemset/problem/122/A

解题思路：因为输入的数据只在 1000 以内，所以我们只需要找到 1000 以内的全部 lucky number，检验是否是他们的倍数即可。

```
i=int(input())
if i%4==0 or i%7==0 or i%44==0 or i%47==0 or i%74==0 or i%77==0
or i%444==0 or i%447==0 or i%477==0 or i%474==0 or i==744 or
i==747 or i==774 or i==777:
    print('YES')
else:
    print('NO')
```

19944:这一天星期几 v0.2, cs10119 Final Exam
math, http://cs101.openjudge.cn/practice/19944/
解题思路：按照给的公式进行推导，注意 1 月 2 月需要先在年份-1 再切片。注意要细心，第一遍把 c 和 y 看反了，第二遍把 m+1 先取整了……所以简单题还改了很多遍，下次要多注意细节。

```
n=int(input())
dic={0:'Sunday',1:'Monday',2:'Tuesday',3:'Wednesday',4:'Thursday'
,5:'Friday',6:'Saturday'}
for i in range(n):
    l=input()
    year=int(l[0:4])+(int(l[4:6])-3)//12
    c=int(str(year)[:2])
    y=int(str(year)[2:])
    m=int(l[4:6])-12*((int(l[4:6])-3)//12)
    d=int(l[6:8])
    print(dic[(y+y//4+c//4-2*c+(26*(m+1))//10+d-1)%7])
```

19949:提取实体 v0.2, cs10119 Final Exam
string, http://cs101.openjudge.cn/practice/19949/
解题思路：先去除所有的连续实体之间的“### ###”，再数剩余的“###”个数就可以。但是依旧注意最后输出的是整个文档的实体个数，不是分行输出！

```
n=int(input())
s=0
```

```
for i in range(n):
    l=input()
    l1=l.replace('### ###',' ')
    s+=l1.count('###')//2
print(s)
```

1017:装箱问题

greedy, http://cs101.openjudge.cn/practice/1017/

解题思路：先装 6*6、5*5、4*4、3*3 的箱子，2*2 先填空，1*1 最后填空。其实本题不需要过多的分类讨论，只需要讨论 2*2 箱子可不可以将 4*4 和 3*3 箱子的空隙填满；如果都填满了，那么 2*2 需要新开箱子，最后填空 1*1 的箱子即可。这里为了减少条件结构，写了一个字典结构、利用最大最小值控制输出。

```
l=input()
while l!='0 0 0 0 0 0':
    a,b,c,d,e,f=map(int,l.split())
    d1={0:0,1:5,2:3,3:1}
    s=d+e+f-(-c)//4
    b1=max(0,b-5*d-d1[c%4])
    s=s-(-b1)//9-min((-4*b-9*c-16*d-25*e-36*f+36*(s-(-b1)//9)-
a),0)//36
    print(s)
    l=input()
```

//选做：

本周的题目只要思路想清楚，遍历是不会超时的。但是我由于 greedy 算法的了解不多，所以不太容易想到思路（其实是不能确定这样遍历是不是一定是最优）。两道题做下来，觉得贪心算法的重点是要从第一个元素出发想到一个局部最优，再确定按照这个思路遍历可否达到全局最优。

16528:充实的寒假生活, cs10117 Final Exam

greedy, http://cs101.openjudge.cn/practice/16528/

解题思路：

参考了一下老师给的思路，考虑第一个选择的活动，一定是结束时间最早的活动，然后再看比这个活动开始时间晚、且结束时间最早的活动，以此类推。对于同一个结束时间的活动只能参加一个，因此也无需对开始时间做排序，并且这种筛选方法可以保证我们取到最优。这里采取二维数组结构，先根据结束时间排序，再在新的列表中选取满足不重合的活动，一次遍历并计数就可以得到答案。不过这里我添加了一个细节，筛选事件的结束时间必须小于等于 60.

解题代码：

```
n=int(input())
l=[]
for i in range(n):
```

```
    l.append([int(x) for x in input().split()])
l.sort(key=lambda x:x[1])
s=0
if l[0][1]<=60:
    s=1
t2=l[0][1]
for i in range(n):
    if l[i][0]>t2:
        s+=1
        t2=l[i][1]
    if l[i][1]>60:
        break
print(s)
```

545C. Woodcutters

dp/greedy, 1500, https://codeforces.com/problemset/problem/545/C

解题思路：

很显然边上的两棵树可以放置在两边，这样的放置方法并不会导致其他的树被阻挡。对于中间的树，我们让它们尽量向左侧倒，如果不能向左倒，那么如果能向右倒就倒，并且更新一下两点之间的距离；这样对于整个列表遍历一遍计数就可以得到结果。另外本题还有一个小细节，如果只有一颗树，那么只需要输出 1，之前默认两边的树都砍掉从 2 开始计数是不对的。

这样做能够达到最优的原因是因为每一个区间只有 3 种可能方式，一种是只能有 1 颗树占据，一种是 2 颗树都可以摆开，还有一种是左右的树过高而都无法占据。1、如果一直向左倒，我们可以保证这是最优解；2、如果有一颗树无法向左倒但是只能向右倒，如果下一颗树还能够向左倒，那么也没有影响最优；3、如果下一颗树只能向右倒，如果之后的树有一颗树能向左倒，那么在这些区间内都有树，回到初始情况，没有影响最优；4、之后的有一颗树不能向右倒，只能保留，那么在这个区间内必然有一颗树无法砍掉，没有影响最优，且回到初始情况；5、每一颗树都只能向右倒，每一颗树都被砍掉，依然是最优。

总而言之，我觉得贪心算法并不需要统筹考虑全局，而是需要想清楚一个局部最优。没有思路的情况下可以先从构造几种典型情况入手来考虑一个算法是不是可以达到局部最优。

解题代码：

```
n=int(input())
if n==1 or n==2:
    print(n)
else:
    l=[]
    for i in range(n):
        l.append([int(x) for x in input().split()])
        s=2
    for i in range(n-2):
        if l[i+1][0]-l[i][0]>l[i+1][1]:
```

```
        s+=1
    else:
        if l[i+2][0]-l[i+1][0]>l[i+1][1]:
            s+=1
            l[i+1][0]+=l[i+1][1]
print(s)
```