Задание 1
Принести грибы

ограничение по времени на тест
20 секунд
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод
стандартный вывод

Кубик пошел по грибы. Он не очень сильный, а грибы очень тяжелые. Кубик может нести не более двух грибов суммарного веса не более K. В процессе прогулки по лесу он нашел N грибов весами $a_1, a_2, \ldots, a_N$.
Определите максимальный вес грибов, который Кубик сможет принести домой, или 0 если он ничего не принес.

Входные данные:
Первая строка содержит два целых числа K, N ($0<K<10^6$, $0<N<10^4$) – максимальный вес который может нести Кубик и количество найденных грибов.
Вторая строка содержит N целых чисел $a_1,a_2,\ldots,a_N$ ($1\le a_i\le 10^6$), где a_i — это вес i-го гриба.

Выходные данные:
Выведите максимальный вес грибов, который Кубик сможет принести домой, или 0 если он ничего не принес.

Пример
входные данные
13 5
18 3 22 9 14
выходные данные
12

Решение на Python:

```
k,n = map(int, input().split())
a = [int(x) for x in input().split() if int(x)<=k]
a=sorted(a)
m=a[-1]
for i in range(len(a)):
    for j in range(i+1,len(a)):
        if a[i]+a[j]>k:
            break
        m=max(m,a[i]+a[j])
print(m)
```

Задание 2
Сколько влезет грибов

ограничение по времени на тест
3 секунды
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод

Кубик взял корзинку и пошел по грибы. Он очень сильный, но корзинка не очень вместительная. Кубик сможет донести все грибы, которые поместились в корзинку. Суммарный объем грибов, которые помещаются в корзинку, не может превышать V. Каждый гриб характеризуется двумя значениями (v, m), где v – объем гриба, а m – вес гриба. Кубик старается брать самые тяжелые грибы. В процессе прогулки по лесу он нашел N грибов (v_1, m_1), (v_2, m_2), …, (v_N, m_N).

Определить максимальный вес грибов, которые Кубик может собрать в корзину.

Входные данные:
Первая строка содержит два целых числа V, N ($0<V<10^5$, $1<N<10^4$) – объем корзинки и количество найденных грибов.
Вторая строка содержит N пар целых чисел грибов (v_1, m_1), (v_2, m_2), …, (v_N, m_N) ($1 \le v_i, m_i \le 10^5$), где v_i, m_i — это объем и вес i-го гриба.

Выходные данные:
Выведите максимальный суммарный вес грибов, которые поместятся в корзину.

Пример
входные данные
10 5
2 5
2 4
2 6
6 8
7 7
выходные данные
19

Решение на Python:

```
v,n = map(int, input().split())
```

```
weight={}
for i in range(n):
    x,y=map(int,input().split())
    if x<=v:
        if weight.get(y)==None:
            weight[y]=[x]
        else:
            weight[y].append(x)
a=weight.keys()
a=sorted(a,reverse=True)
answer=0
for x in a:
    m=sorted(weight[x])
    for y in m:
        if y>v:
            break
        v-=y
        answer+=x
print(answer)
```

Задание 3
Сходить гости

ограничение по времени на тест
3 секунды
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод

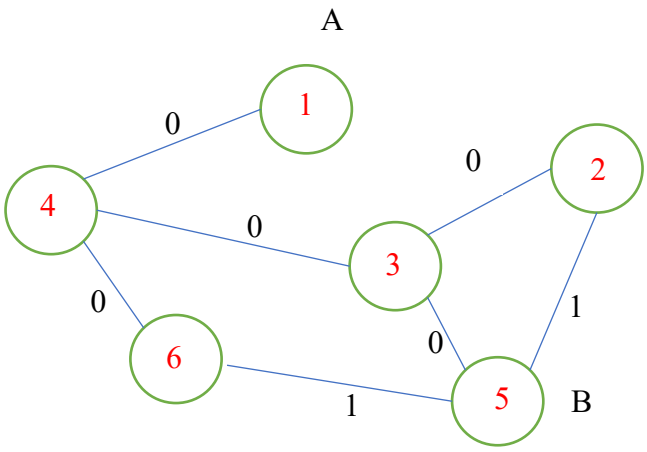
Кубик и Кубичка живут в лесу. Их дома располагаются на полянках. Передвигаться по лесу можно только по тропинкам, которые соединяют полянки. Кубик решил сходить в гости к Кубичке. Гарантируется, что Кубик сможет прийти к Кубичке в гости двигаясь по тропинкам. Известно, что, проходя по некоторым тропикам Кубик испачкается.
Определите сможет ли Кубик прийти в гости к Кубичке чистым.

Входные данные:
Первая строка содержит одно целое число n и m ($2 \le n \le 500;\ 1 \le m \le 5000$) – количество полянок и количество тропинок.
Вторая строка содержит два целых числа A, B – номера полянок, на которых живут Кубик и Кубичка.
Затем следует m строк, в каждой строке записано три целых числа: p, r, k ($1 \le p,r \le n,\ 0 \le k \le 1$)— номера полянок, между которыми есть тропинка и признак испачкается или нет Кубик на этой тропике (k равно 1 если Кубик испачкается и равно 0 в противном случае).

Выходные данные
Выведите YES если Кубик сможет прийти в гости к Кубичке чистым и NO в противном случае.

Примеры
входные данные
```
6 7
1 5
1 4 0
4 6 0
4 3 0
2 3 0
6 5 1
3 5 0
5 2 1
```
выходные данные
```
YES
```

Примечание



Решение на Python:

```
n,m = map(int, input().split())

A,B = map(int, input().split())
A-=1
B-=1
neighbours=[[] for _ in range(n)]
for i in range(m):
    g1,g2,k = map(int, input().split())
    neighbours[g1-1].append(tuple([g2-1,k]))
    neighbours[g2 - 1].append(tuple([g1-1,k]))
v=[A]
for x in v:
    for y in neighbours[x]:
        if y[1]==0 and (y[0] not in v):
            v.append(y[0])
if B in v:
    print('YES')
else:
    print('NO')
```

Задание 4
Красивый букет для Кубички

ограничение по времени на тест
7 секунд
ограничение по памяти на тест
10 мегабайт
ввод
стандартный ввод
вывод

Кубик решил подарить Кубичке букет цветов. Кубичка считает букет красивым, если он состоит из не более, чем двух видов цветов, а количество цветов в букете кратно семи. Кубик собирал цветы на полянках. Известно, что на каждой полянке Кубик либо собирал цветы только одного вида, либо не собирал их вовсе. Всего в лесу N полянок. На i-ой полянке можно собрать a_i тюльпанов, b_i ромашек, c_i лютиков и d_i васильков.
Определите максимальное количество цветов в красивом букете, который Кубик может подарить Кубичке.

Входные данные:
Первая строка содержит одно целое число n ($0 \le n \le 10^6$) – количество полянок
Затем следует n строк, в каждой строке записано четыре целых числа: a, b, c, d ($1 \le a,b,c,d \le 10^4$)— количество тюльпанов, ромашек, лютиков и васильков.
.

Выходные данные
Выведите целое число – максимальное количество цветов в букете.

Примеры
входные данные
5
1 5 8 6
9 4 7 3
4 6 2 4
4 3 1 5
8 1 6 7
выходные данные
28

Решение на Python:

```
n = int(input())

ab=0
ac=0
ad=0
bc=0
bd=0
```

```
cd=0

for i in range(n):
    a,b,c,d = map(int, input().split())
    ab += max(a,b)
    ac += max(a,c)
    ad += max(a,d)
    bc += max(c,b)
    bd += max(d,b)
    cd += max(c,d)
m=max(ab, ac, ad, bc, bd, cd) //7
print(m*7)
```

Задание 5
Спасти Кубичку

ограничение по времени на тест
20 секунд
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод

Кубик и Кубичка живут в лесу. В лесу случился потоп, и он превратился во множество островов. Кубик и Кубичка оказались на разных островах. На каждом острове растут деревья. Кубик увидел, что есть деревья, высота которых позволит дотянуться до другого острова если их срубить, и тогда Кубик сможет перебраться с одного острова на другой. К сожалению, у Кубика очень старый инструмент, который позволяет срубить только 5 деревьев.
Определите сможет ли Кубик добраться до острова Кубички.
Если высота дерева превышает расстояние между островами, то считать возможным перебраться по этому дереву с одного острова на другой, иначе – нет.

Входные данные:
Первая строка содержит одно целое число n ($2 \le n \le 500$) – количество островов.
Вторая строка содержит два числа A и B – номера островов, на которых оказались Кубик и Кубичка.
Следующие n строк содержат по n чисел $r_{i1}, r_{i2}, \ldots, r_{in}$ ($0 \le r_{ij} \le 10^5$, $r_{ii}=0$) – расстояния между островами (r_{ij} – расстояния от i-го острова до j-го острова).
Следующие n строк содержат список высот $l_{i1}, l_{i2}, \ldots, l_{im}$ ($1 \le l_{ij} \le 10^5$, $0 \le m$) деревьев на расположенных на островах (i-я строка содержит высоты деревьев на i-м острове).

Выходные данные:
Если Кубик сможет добраться до острова Кубички, выведите «YES». Иначе выведите «NO».

Примеры
входные данные

```
5
1 5
0 4 2 6 5
4 0 2 6 2
2 2 0 5 3
6 6 5 0 8
5 2 3 8 0
1 2 3 5
2 3
1 2 4 6
```

2 5 3
1 2

выходные данные
YES

Примечание
От первого острова до второго понадобиться дерево высоты 5, от второго до третьего – высоты 3, от третьего до пятого – высоты 4.

Решение на Python:

```
n = int(input())
A,B=map(int,input().split())
A-=1
B-=1
r=[]
l=[]
for i in range(n):
    r.append([int(x) for x in input().split()])
for i in range(n):
    l.append(max([int(x) for x in input().split()]+[0]))
island=[]
def search(island):
    if len(island)>5:
        return False
    if len(island)==5 or island[-1]==B:
        return island[-1]==B
    for i in range(n): #очередной остров
        if i in island:
            continue
        if r[island[-1]][i]<l[island[-1]]:
            isl=island.copy()+[i]
            if search(isl):
                return True
    return False
if search([A]):
    print("YES")
else:
    print("NO")
```

Задание 6
Построить ограду

ограничение по времени на тест
20 секунд
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод

На участке Кубика растут N деревьев. Дерево с номером i имеет координаты (x_i, y_i). Кубик решил огородить свой участок соединяя забором. Забор не должен приближаться к дереву ближе, чем на 1 метр. Таким образом, каждое дерево на участке Кубика находится внутри участка.
Найдите минимальную длину забора с точностью до двух знаков, который нужно построить Кубику.
Масштаб координатной сетки 1:1. Гарантируется, что есть три дерева, не лежащие на одной прямой.

Входные данные:
Первая строка содержит одно целое число n ($3 \le n \le 2000$) – количество деревьев. Затем следует n строк, в каждой строке записано два целых числа: x, y ($-10^4 \le x, y \le 10^4$)— координаты дерева.

Выходные данные:
Выведите число с точностью до двух знаков после десятичной запятой– минимальная длина забора.

Примеры
входные данные
```
6
0 0
2 2
1 2
0 3
3 0
3 3
```
выходные данные
```
15.14
```

Решение на Python:
```
import math
n = int(input())
tree=sorted([tuple(map(int,input().split())) for i in range(n)])
fence=[tree[0]]
```

```
def onFence(k):
    if  k in fence:
        return True
    if len(fence)<2:
        return False
    for i in range(1,len(fence)):
        x=fence[i-1]
        y=fence[i]
        if (k[0] - x[0]) * (y[1] - x[1]) - (k[1] - x[1]) * (y[0] - x[0]) == 0:
            return True
    return False

for x in fence:
    for y in tree:
        if onFence(y):
            continue
        r=((x[0]-y[0])**2+(x[1]-y[1])**2)**0.5
        nexttree=y
        for k in tree:
            if (k[0]-x[0])*(y[1]-x[1])-(k[1]-x[1])*(y[0]-x[0])!=0:
                if (k[0]-x[0])*(y[1]-x[1])-(k[1]-x[1])*(y[0]-x[0])<0:
                    sign=-1
                else:
                    sign=1
                break
        p=True
        for k in tree:
            if ((k[0]-x[0])*(y[1]-x[1])-(k[1]-x[1])*(y[0]-x[0]))*sign<0:
                p=False
                break
            if ((k[0]-x[0])*(y[1]-x[1])-(k[1]-x[1])*(y[0]-x[0]))==0:
                if ((x[0]-k[0])**2+(x[1]-k[1])**2)**0.5>r:
                    r = ((x[0] - k[0]) ** 2 + (x[1] - k[1]) ** 2) ** 0.5
                    nexttree = k
        if p:
            fence.append(nexttree)
            break
x=fence[0]
y=fence[-1]
l=math.pi+((x[0]-y[0])**2+(x[1]-y[1])**2)**0.5
for i in range(len(fence)-1):
    x = fence[i]
    y = fence[i+1]
    l += ((x[0] - y[0]) ** 2 + (x[1] - y[1]) ** 2) ** 0.5
print(round(l,2))
```